Object Detection and Recognition with Event Driven Cameras

Author: Massimiliano Iacono

Supervisors: Dr. Chiara Bartolozzi
Dr. Arren Glover
Prof. Emre Neftci
Dr. Vadim Tikhanoff

IIT - Istituto Italiano di Tecnologia

University of Genova

PhD Thesis for the curriculum of

Bioengineering and Robotics

April 15, 2020

Acknowledgements

I would like to thank everyone who have supported me during these years.

My main supervisor, Chiara, who has always motivated me, not only to pursue my academic goals, but also to enrich myself both personally and professionally, with some of the most unforgettable experiences I could take part in.

My everyday guide, Arren, who helped me improve my methodology, coding skills and overall scientific soundness of my research.

Emre, who hosted me in California and allowed me to live an amazing and interesting part of my PhD.

Vadim, who has always made his valuable expertise available with precious advises.

My colleagues, Marco, Fabrizio, Giulia and all the rest of the iCub Team for both moral and technical support, particularly during our countless coffee breaks.

My girlfriend Letizia, for her continuous presence by my side, especially for bearing me during the writing enclosures for articles and thesis.

My family, for allowing me to pursue my studies both in Italy and abroad and always pushing me to get the maximum out of myself.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Massimiliano Iacono

April 15, 2020

Abstract

This thesis presents study, analysis and implementation of algorithms to perform object detection and recognition using an event-based camera. This sensor represents a novel paradigm which opens a wide range of possibilities for future developments of computer vision. In particular it allows to produce a fast, compressed, illumination invariant output, which can be exploited for robotic tasks, where fast dynamics and significant illumination changes are frequent. The experiments are carried out on the neuromorphic version of the iCub humanoid platform. The robot is equipped with a novel dual camera setup mounted directly in the robot's eyes, used to generate data with a moving camera. The motion causes the presence of background clutter in the event stream.

In such scenario the detection problem has been addressed with an attention mechanism, specifically designed to respond to the presence of objects, while discarding clutter. The proposed implementation takes advantage of the nature of the data to simplify the original proto-object saliency model which inspired this work.

Successively, the recognition task was first tackled with a feasibility study to demonstrate that the event stream carries sufficient information to classify objects and then with the implementation of a spiking neural network. The feasibility study provides the proof-of-concept that events are informative enough in the context of object classifi-

cation, whereas the spiking implementation improves the results by employing an architecture specifically designed to process event data. The spiking network was trained with a three-factor local learning rule which overcomes weight transport, update locking and non-locality problem.

The presented results prove that both detection and classification can be carried-out in the target application using the event data.

Contents

List of Figures

Chapter 1

Introduction

This thesis will focus on object detection and recognition using event driven cameras.

Event driven cameras represent a great potential for the future of computer vision, given their unprecedented capability to deliver a compressed output at a very fine temporal resolution and high dynamic range. With these characteristics they unlock a whole new range of possibilities, especially in robotics, where the scenarios may include fast dynamics and large variations in illumination. Additionally, they work on much lower power requirements as compared to regular frame-based sensors, opening the possibilities for embedded applications in power constrained situations resulting in a higher sustainability of robots with an onboard processing, increasing their battery life.

The neuromorphic community is working hard to understand the advantages and limitations of this new paradigm, in any sort of visual task, trying to push the boundaries of the field of applicability of the sensor. Chapter 2 gives an overview of the state of the art in the field, also providing examples of how this sensor has captured attention even out of the academic world.

The present study aims at understanding the possibility of employing the sensor for the purpose of object detection and recognition in a robotic scenario. The robotic platform where all of the experiments are carried out is the neuromorphic version of the iCub humanoid platform [1, 2], equipped with two event cameras in the eyes, coupled with regular cameras, as further described in Section 4.1.1.

The work is motivated by the need of modern computer vision systems to expand their range of applicability. Standard cameras suffer from several limitations, such as limited frame-rate, poor image quality in case of over or underexposure, high memory space consumption. On the other hand, being a relatively new technology, the algorithms developed for event cameras are not as mature as their frame-based counterpart. For this reason the community needs to invest resources to fill this technological gap, and the present work represents a step towards that direction.

Both object detection and recognition tasks have been widely studied in the traditional computer vision field, especially since the rise of deep learning, and have started to gain interest also in the neuromorphic community. However, there is a significant difference between the output of the two sensors that limits the use of existing algorithms designed for RGB images. In fact, the event-camera only responds to changes in the stimulus rather than its absolute appearance. They elicit a spike every time an "event" is detected, an event being a relative change in the light intensity falling on a pixel. More details about the event cameras and the commercially available products will be provided in Section 2.1.

When approaching the problem building on a traditional computer vision

starting point there are several possibilities to develop an algorithm for neuromorphic cameras, which could be generally categorized as (i) direct application of the existing method, (ii) adaptation by changing the incompatible parts of the algorithm and (iii) the development from scratch. Each approach has strengths and weaknesses. (i) can be mainly used as an exploratory technique for a feasibility study or a proof-of-concept, not taking into account the sensor capabilities, but still useful to understand how to proceed in further development. The use of approach (ii) gets closer to an algorithm design that takes advantage of the events and it may result in algorithmic improvement over the frame based approach, however to fully exploit the sensor characteristics method (iii) offers the best results at the cost of increased design and development time. This thesis will demonstrate various applications of all of these different approaches, which will be briefly introduced in the following paragraphs.

The first algorithm is presented in Chapter 3, entitled *Proto-object based saliency for event cameras*. The proposed approach is a biologically plausible object detection system based on a visual attention mechanism. In this case the algorithm was adapted from an existing model [3] by modifying the processing stages that proved incompatible with the event data, therefore following approach (ii). The use of event cameras helped, on an algorithmic point of view, to simplify the computation resulting in a lighter implementation. In particular, as discussed in more details in Chapter 3, [3] uses a Center Surround (CS) filtering that in the event based implementation is already performed at the sensor level, allowing a significant simplification of the algorithm. Furthermore, the neuromorphic camera increases the overall biological plausibility of the system by mimicking the early computation stages of the biological visual processing pipeline.
The attention mechanism forms the basis of the object detection system by se-

lecting regions of the visual field which most likely contain objects and therefore represents the first stage of a more complete image processing pipeline. This work was presented at the International Conference on Intelligent Robots and Systems (IROS) 2019 conference [4].

Chapter 4 introduces a direct application of an off-the-shelf deep learning architecture for the purpose of object recognition. This work presents a proof-of-concept of events usability for the target task, therefore approach (i) has been employed. The goal was to prove that the events contain the informations necessary to extract features that are discriminative enough to classify different objects. The question raised from the inherent difference of event appearance as compared to RGB images. At first sight, events seem to contain much less information, lacking colours, intensity, or fine texture details contained in traditional images. Nevertheless, the presented analysis showed how the information contained in the event stream allows to achieve a recognition performance in par, if not slightly better, than a traditional camera under the same settings. Additionally, an ad-hoc experiment proved that the event cameras can still perform the recognition under extreme conditions such as very high object speed or poor illumination. The latter result demonstrates how the event cameras can compensate for the failures of traditional recognition systems. In order to use the events in a frame-based algorithm, the data must be represented in an image form. To achieve this events are accumulated over a temporal window and this work also analyses the sensitivity of the system to the length of the integration time, providing some guidelines for proper tuning of the temporal parameter.
One of the main limitations when applying machine learning to event data is the lack of annotated dataset as opposed to the RGB domain. For this reason this work also exploited the hardware setup mounted in the iCub's eyes in order to

collect an annotated dataset used for training the system. The data labelling was performed in an automatic way, bootstrapping recognition algorithms on frames. Results were presented in IROS 2018 conference [5].

The latter work proved that events contain the information necessary for a recognition system to work, however it does not really take advantage of the event camera capabilities, because when rendering an image at a fixed frame-rate the rich temporal information is lost. For this reason, Chapter 5 analyses a learning framework that is specifically designed to process event data, following approach (iii). In the proposed implementation, a Spiking Neural Network (SNN) is trained with a novel technique, Deep Continuous Local Learning (DeCoLLe), which allows the exploitation of already available machine learning software tools, still taking in consideration the nature of the data and the biological plausibility of the learning rule. DeCoLLe achieves higher classification accuracy then the adapted off-the-shelf method proposed in Chapter 4, where a state of the art deep architecture was employed. This result is interesting because the DeCoLLe learning rule relaxes most of the constraining assumption made by the ubiquitous Back Propagation (BP) algorithm such as symmetric synaptic weights (weight transport problem), use of high precision error signal, alternation of forward and backward passes (update locking problem). Additionally, the spiking network could eventually be deployed on dedicated neuromorphic hardware for a low-latency, power efficient and biologically plausible implementation. Chapter 5 will provide more details about the derivation of the learning rule and the architecture in general.
This work was the result of a collaboration with the Neuromorphic Machine Intelligence (NMI) laboratory of University of California, Irvine under the supervision of professor Emre Neftci.

This collaboration is still active, with the aim of integrating the attention and recognition modules together for a full detection and classification pipeline.

Chapter 2

State of the Art

This chapter aims at summarizing the latest research results that make use of event driven cameras, with particular focus on works addressing object detection and recognition. For this task, the computer vision community has quite rapidly switched to learning approaches, with the aid of more sophisticated architectures, more powerful hardware and the availability of large amount of data. Machine learning, and especially deep learning, have changed the way computer vision algorithms are developed, due to their increased precision, versatility and variety of applications to which they are applied [6]. In parallel, as further explained in Section 2.1, the rise of event based cameras such as Dynamic Vision Sensor (DVS) [7], DAVIS [8] or Asynchronous Time-based Image Sensor (ATIS) [9] has unlocked new possibilities, by filling the gaps where regular cameras usually fail, due to their high temporal resolution, high dynamic range and low power consumption. In this scenario, it seems inevitable that also the neuromorphic community would make use of deep learning algorithms using data taken from event cameras. One possible approach is the use of off-the-shelf deep learning architecture that have been developed for frame-based data and simply feed them with events, as presented in Chapter 4. However, there are several differences between frame-based and event-based sensors that need to be taken into account when designing an

algorithm that takes events as input.The data comes in a very different way as compared to regular cameras, in which a full matrix of data with RGB values is sent out at a fixed frame rate. Event cameras only detect the change in the light intensity falling at each pixel producing a fast, asynchronous, compressed data stream, that inherently contains informations such as local contrast (useful in edge detection), or fine temporal scale for high speed applications. Such sensor allows for the extraction of richer spatio-temporal information, and enabling a data-driven computing approach that significantly differs from the batch processing of regular cameras. Clearly, this fundamental difference with respect to traditional sensors requires some considerations, not only when working with machine learning, but with any algorithm that takes data from event cameras as input: while frame-based sensors output a full matrix of RGB values at a fixed frame rate, event cameras output info as a continuous stream of events, each corresponding to an illumination change in the field of view of the pixel that emitted the event. Section 2.2 explains the main techniques that have been used in the literature to cope with this type of data.

On the other hand, the spikes produced by a neuromorphic sensor fit within the domain of Spiking Neural Network (SNN) [10], in which the inter-neuron communication is carried out with spikes. These networks, besides being inspired by the most intelligent machine known to date, the human brain, also promise to reduce computational cost and power requirements. To achieve this, computation is only performed when needed, and communication is carried out with binary signals, significantly reducing the amount of memory required as compared to double-precision floating point values typically used in Convolutional Neural Network (CNN)s. Unfortunately SNNs fed with event data are struggling to achieve state of the art performance as compared to deep Artificial Neural

Network (ANN) due to the lack of effective learning algorithms, because the non differentiability of spikes hinders the direct application of backpropagation. Chapter 5 presents a SNN implementation that overcomes this problem and also relaxes the strong assumptions of BP. Section 2.3 provides an overview of the latest techniques available in the literature to effectively train networks of spiking neurons.

To take full advantage of the capabilities of SNNs, the community has developed a number of hardware devices that can emulate on-chip the behaviour of spiking neurons. As compared to Graphics Processing Unit (GPU)s, that aim at providing general purpose acceleration of graphic processing, the neuromorphic hardware is more optimized towards the deployment of neural network, with massive parallelization and minimum communication overhead. These devices tend to keep memory and processing in the same place, in order to circumvent the bottleneck that traditional von Neumann architectures face [11]. However, the deployment on neuromorphic hardware requires a proper architectural design of the algorithms. The system presented in Chapter 5 is designed taking in consideration the hardware implementation and an ongoing project is focusing on porting the attention system introduced in Chapter 3 to a SpiNNaker device. Section 2.4 will list some of the most promising neuromorphic chips with further details on the architecture of each.

As previously mentioned, one of the main reasons that has made deep learning so successful is also the huge amount of annotated datasets available for training. Also in this sense, the neuromorphic community has a big challenge to face, but in the recent past, more and more datasets are becoming available to benchmark algorithms. Section 2.5 lists some of the most interesting ones, highlighting

those that are currently available to the public. Chapter 4 also presents a dataset recording with an automatic annotation system that exploits the hardware setup, comprising both event and regular cameras, mounted on the iCub robot.

Finally Section 2.6 provides more details about the applications of event driven cameras to the object detection and recognition task.

In particular, in order to develop an algorithm that better suits the event data, Section 2.6.1 focuses on how to achieve object detection with a bottom-up attention mechanism, as further explained in Chapter 3.

Instead Section 2.6.2 gives a literature overview of works that address the object classification task starting from event data. In this thesis Chapters 4 and 5 will provide two different approaches using a frame-based or spiking architecture.

2.1 Event-driven cameras

Event driven cameras, otherwise called silicon retinae as in their first development [12], are an emerging technology that represent a significant revolution as compared to traditional vision sensors. They can operate on an unprecedented temporal resolution, in the order of microseconds, at a very low power budget. Each pixel is an independent illumination change detector, that asynchronously produces a spike whenever the change in the logarithmic light intensity falling on it is significantly large, i.e. at the edges of objects or structures moving relative to the camera. The logarithmic scale is given by the response of the photoreceptor used at the circuitry level. The *event-stream* does not encode redundant data and, in the case of dynamic vision sensors, when no motion or illumination change occurs there is simply no events to process. Uniformly textured surfaces do not elicit a response, even during fast motion. The result is a compressed,

asynchronous visual stream with very high temporal resolution ($< 1\mu s$). Additionally, as compared to the traditional dynamic range of a regular camera (60-80 dB), the event cameras provide a much higher dynamic range ($120 - 140dB$), as well as a low latency ($< 15\mu s$ vs $30ms$ of traditional cameras) as a full frame of data does not need to be transferred in unison. The sensors have great potential for use in dynamic robotic tasks as they would cover all the cases where regular cameras would fail, such as poor lighting conditions, fast motion of camera or objects/scene. One famous case of such failure recently occurred when a camera on an Uber self driving car has failed in detecting the presence of a pedestrian at night [13]. In such scenarios, the event cameras could have detected the pedestrian. In recent years, the research community has shown an increasing interest towards this technology with applications ranging from estimation of depth and/or optical flow [14–17], human pose estimation [18], gesture recognition [19–22], mapping [23, 24]. Since the first development in 1991 [12], and later attempts [25, 26], the technology has matured with the development of the first Dynamic Vision Sensor (DVS) at IniLabs [7, 27, 28]. The DVS, shown in Figure 2.1 is a 128×128 resolution camera which has all of the characteristics described above (high temporal resolution, high dynamic range, low latency). In particular the DVS was the first device that came along with an easy-to-use library that made it practical for non-expert users, and that overcame the pixel mismatch problem, which led to high variance in pixel firing rates in previous development (i.e. 1–2 decades standard deviation and more than half the pixels not spiking at all for stimuli with 50% contrast [26]). Therefore, one key DVS feature is contrast sensitivity. In [29] a novel pixel photo sensing and transimpedance pre-amplification stage makes it possible to improve even further the contrast sensitivity from 10-15% down to 1.5%.In a successive development the DAVIS240 combined both a DVS, with a slightly increased spatial resolution of 240×180,

Figure 2.1: A DVS camera

and a regular camera to provide an overlapped output of both greyscale intensity images and events [8,30]. The Inivation company [31], spin-off of IniLabs, is now manufacturing and selling both the DVS and the DAVIS

Samsung [32] and Insightness [33] have also started developing products based on the DVS. Interestingly Samsung has also released the first commercial product to the public: an indoor home monitoring system that exploits the low power consumption, data compression and motion detection of the sensor [34].

At the same time the ATIS sensor was also being developed [9], with a resolution of 304×240 and the possibility of reconstructing intensity images exploiting electrical properties of the pixel circuit at a cost of an increased pixel size as compared to DVS or DAVIS. Based on the ATIS new generation high resolution

technology the Prophesee company is currently producing and selling event based sensors and cameras [35].

Overcoming the limited resolution of mentioned sensors the Celex company has been able to develop an outstanding 1280×800 resolution event-based camera which unlocks even new possibilities for algorithm deployment.

In general, also on a commercial and industrial point of view, the sensor is attracting interest towards the development of new generation vision tools.

2.2 Data representation

When dealing with event based cameras, the first thing that needs to be defined is how to represent the data. Event cameras do not output a whole matrix of pixels, but a stream of events, typically arranged in an array in Address Event Representation (AER) [36]. In AER, each event contains information about the pixel that has fired, together with a timestamp and the event polarity (positive if light intensity on the pixel has increased, and negative otherwise). In a multi camera setup the event can also contain an ID of the camera. Such a compact and sparse representation allows for data compression, but it has to be treated differently than regular RGB matrix representation. In literature there are a number of different approaches to deal with AER, that must be carefully analysed before choosing the one that better suits the desired task and algorithm. Especially when designing a neural network that takes events as input, the data representation plays a crucial role for the correct functioning of the network. Since most of the computer vision algorithms are designed to work with RGB images, the most straightforward thing to do is to render an image from the event stream. This step is only needed in synchronous algorithms, such as those presented in

Chapters 3 and 4, that are directly adapted from the computer vision domain. In general, the individual spike does not contain enough information, so events need to be integrated in order to generate a more meaningful representation.

There are several ways to produce an image from asynchronous events, but the most simple is the window approach where events are accumulated over a fixed amount of time [37], or over a fixed number of events [18, 38, 39]. The first approach is agnostic to the type of scene, but it may suffer from motion blur or incomplete edge representation depending on object or camera speed. The second approach automatically adapts the frame rate according to the amount of motion, hence it is less sensitive to blur, but it requires some assumptions on the number of events that the camera would generate in a given environment, depending on clutter, texture and number of objects in the scene. Figure 2.2 gives an idea of the times at which frames would be generated in either representation. In the case of a fixed temporal window, each frame comes at a given rate regardless of the amount of events emitted between one frame and the other, whereas in the second representation, the number of generated frames changes depending on the stimulus. Additionally, the per-pixel count of the events could be used as extra information, encoding the amount of activity in a given area, possibly due to high contrast or motion. In general, most of the more sophisticated approaches are variants of these two simple representations. However, both of these approaches discard the rich temporal information embedded in the events, so they do not fully exploit the event cameras potential.

To take this valuable information into account, [40] introduces the time surfaces, where the most recent events get higher values that then decay over time, maintaining an history of recent activity. Some example time surfaces are shown

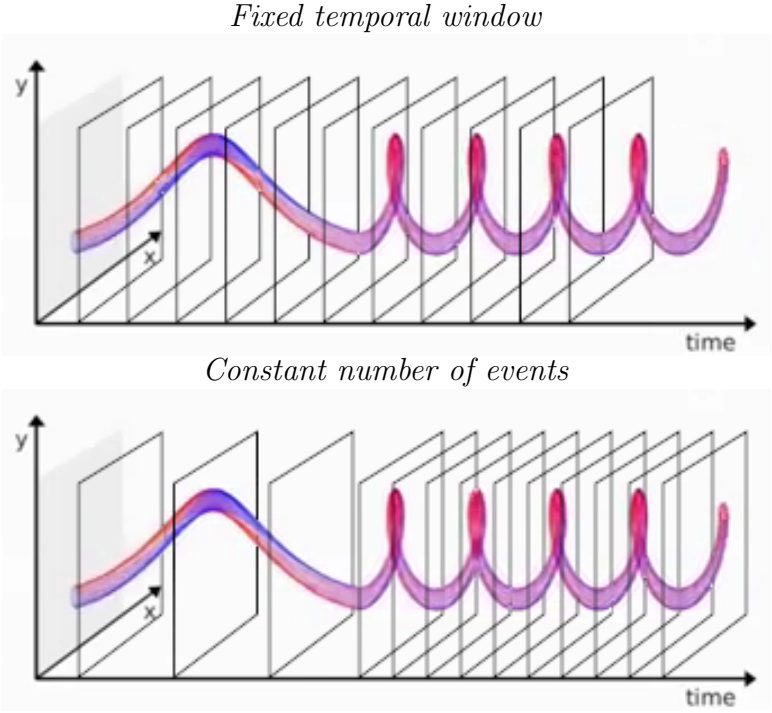


Figure 2.2: Example of fixed temporal window (top) and constant number of events (bottom) representation. In the first case, each frame (black rectangles) comes at a given rate regardless of the amount of events emitted between one frame at the other, whereas in the second representation, the number of generated frames changes depending on the stimulus. The image plane xy is plotted against the time dimension.

in Figure 2.3. In [21] time surfaces are successfully used to perform gesture recognition, a task in which the proper encoding of temporal information is fundamental. Time surfaces can be further enhanced by adding two additional feature maps computed with linear decay and binning (each timestamp is maintained for a fixed amount of time or overridden by a new event at the same location), as introduced in [41]. This new representation is called memory surface, as it encodes different ways to remember/forget past data. In [42] the authors introduce the leaky surface layer, that is a matrix in which each element is connected to a pixel of the camera and every time a spike arrives the corresponding element in the surface is increased by a fixed amount and then the whole matrix decayed

depending on the time from the last spike. The use of this technique allows the exploitation of the rich temporal information contained in spikes and keep track of pixels with high activity. In [43] a time-image representation is introduced, in which they consider the average timestamp of the events occurred at a given pixel in a fixed temporal window. This approach allows for noise mitigation, while maintaining the valuable temporal information. Furthermore a similar representation is presented in [44], however they discretize along the time axis to a fixed number of bins and generate what they call a discretized event volume. Each event falls into the bin corresponding to its timestamp, maintaining a quantized temporal distribution of events over the window. Table 2.1 summarizes the main techniques to represent event data.

Different ways to render an image from events could have a significant impact on the algorithms outcome, as shown in [45] (also explained in Section 2.3), in which a proper choice of data representation was crucial to train an ANN. As an example Chapter 4 will present a study which analyses how the temporal parameter for event integration affects the result of a CNN for object recognition. However, the direct use of spikes in the domain of a SNN allows to skip this step, feeding the raw data in the network. Also in this scenario, information from multiple events need to be integrated by the network. Chapter 5 will go more in the details of data integration in a SNN.

2.3 Training a Spiking Neural Network

Given the data representation of event cameras, the SNN framework [10] seems to be better suited than traditional CNN. In SNNs, the neuron-to-neuron communication is carried out in form of spikes, similarly to what happens on a synaptic level in the brain. As such, the pixels of an event camera can be directly treated

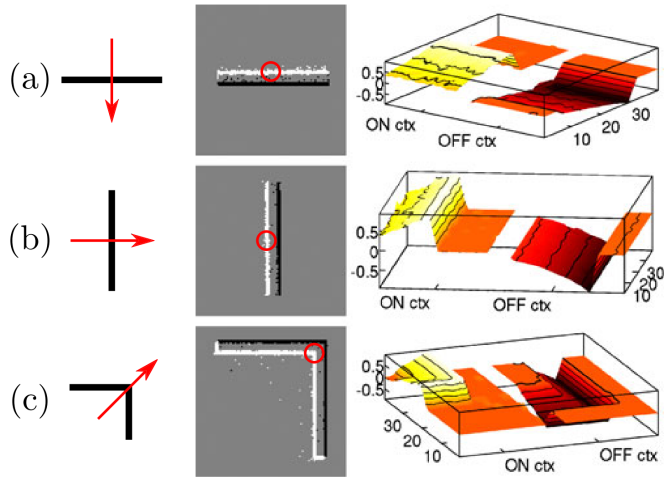


Figure 2.3: From [40]. Example time surfaces when presenting simple moving edges to the camera. First column shows a representation of the stimulus. The second column shows corresponding data from the ATIS sensor where white dots are ON events and black dots are OFF events. The third column shows the time-surface obtained from these events and computed for the event located in the centre of the circle in the second column: the first, positive, half is obtained from the ON events and the second, negative, half is obtained from the OFF events

as one of those neurons which emit spikes to communicate visual information. SNNs have the advantage of being much less expensive in terms of data exchange and power consumption, while providing superior computational power than traditional neural networks [47]. One of the reasons why SNNs have not achieved the success of ANNs is the lack of an effective training procedure: spikes are non differentiable by definition, preventing the gradient from propagating backward as required in the ubiquitous backpropagation algorithm. However, due to the availability of neuromorphic sensors and hardware, the research community has shown interest in finding ways to train networks of spiking neurons. These training techniques can be categorized under three groups: (i) mapping the weights of a pre-trained ANN to a SNN, (ii) use of variant of backpropagation directly on the SNN and (iii) unsupervised techniques which do not require the gradient computation at all.

Data Representation	Comments
Temporal window	Scene independent. May suffer from blur or underexposure. Discards temporal information.
Constant number of events	Adapts to different speed. To be tuned on specific scene. Discards temporal information.
Time surface [40]	Maintains temporal information. May be updated asynchronously.
Memory surface [41]	Enhances time surface. Encodes different ways to remember/forget history.
Leaky surface layer [46]	Maintains temporal information and keeps track of pixels with high activity
Time-image [43]	Average of timestamps over fixed temporal window to reduce noise.
Discretize event volume [44]	Discretizes the temporal dimension, maintaining the distribution of events over time.

Table 2.1: Summary of data representation techniques

The first approach applies regular backpropagation on an ANN and then maps the learned weights on the spiking counterpart. Events are turned into frame using a fixed temporal window representation, and fed to a CNN for training. The weights learnt this way are then transferred to a spiking architecture identical to the one used for training, except that each neuron is replaced with a LIF model [48]. High activation of a neuron in the CNN is therefore translated into a high firing rate of the corresponding spiking one. This algorithm was first presented in [37], then used again in [49] where the authors present promising results on synthetic data, however, as admitted by the same authors, accuracy drops dramatically when the input data are recorded from real sensors. More recent work using the same technique [45] has managed to reduce this performance gap on a robot predator-prey chasing dataset recorded in [50], where a "predator" robot had to recognize and chase a "prey" robot using a DVS. To achieve

this result, the way frames were generated from events to train the ANN turned out to be a crucial point. They have used a constant number of events with per-pixel count. Additionally, outliers are removed by clipping counts greater than 3 times the standard deviation, and finally the pixel values are scaled to the range $[0, 1]$. This approach is easier to implement, given the availability of tools to perform backpropagation and, now, to carry out the weight mapping to a SNN.

A second class of algorithms addresses the problem by finding workarounds and/or approximations of the classical backpropagation approach to overcome the non differentiability of the spikes. One example of such algorithms is the Event Driven Random Backpropagation (eRBP) [51], which in turn is inspired by [52]. The latter shows that there is no necessity of symmetric backward connectivity pattern for an accurate weight update. In the classical backpropagation algorithm the update is computed as $\delta_{bp} = W^T e$ where W is the weight matrix and e the error signal. In [52] instead, W^T is replaced by a matrix of random and fixed weights that only needs to be initialized once. In other words, when performing one gradient descent optimization step, the neurons do not need to know the synaptic weight value and instead receive a random projection of the error signal. This algorithm, called Feedback Alignment, achieves remarkable results matching the same level of performances of backpropagation while requiring a much less constrained memory sharing mechanism. Such a feature is fundamental when deploying a network on neuromorphic hardware where the memory is only available locally at the neuron level. Building on the finding of [52], eRBP introduces an event-driven learning rule that can learn with high accuracy on MNIST [53] and EMNIST [54]. The spike non differentiability is dealt with by means of surrogate gradients, that is an approximated derivative of the hard threshold of spikes on the backward pass only. The use of surrogate gradient serves as a relaxation of

the spiking non linearity for the purpose of numerical optimization, allowing the minimization of the loss function as in the regular backpropagation algorithm. Figure 2.4 shows an example of such relaxation in a case where the gradient of the loss function is flat almost everywhere. [55] provides an overview of several works that make use of surrogate gradients. Out of these techniques, it is worth mentioning the work presented in [20], that makes use of a three-factor learning rule combining surrogate gradient, random backpropagation and local learning [56] on a spiking neural network that performs gesture recognition.

SNNs can also be trained in an unsupervised fashion, making use of Spike Timing Dependent Plasticity (STDP) [57]. Such learning rule is a type of Hebbian rule, named after the neuroscientist Donald Hebb who first theorized that "Neurons that fire together, wire together". In other words connections between neurons that show a causal relationship are strengthened, that is, those which fire right after having received spikes form a presynaptic neuron, while connections of the neurons that show the opposite behaviour are weakened. The increase of synaptic strength is called Long Term Potentiation (LTP), whereas the opposite is called Long Term Depression (LTD). Figure 2.5 illustrates the relation between spike timing and synaptic strength adaptation. The use of this learning rule , allows training of SNN without the need of annotated dataset, making it very interesting for practical applications where annotated data are scarcely available. One example of successful application of STDP can be found in [58], which will be further explained in section 2.6.2. However unsupervised learning struggles in achieving similar accuracy as compared to supervised, and in general has more complex dynamics which are hard to regularize.

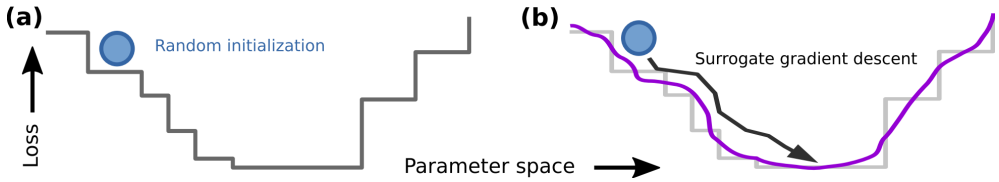


Figure 2.4: From [55]. Example of a loss function with vanishing gradient everywhere (left) and how the use of surrogate gradient may help smoothing to support the optimization (right)

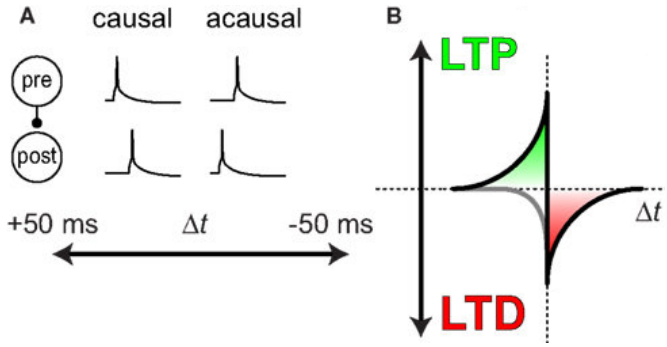


Figure 2.5: On the left an example of pre and post synaptic neurons that show causality, in that the post fires right after the pre, or vice versa. In the first case the connection between the two will be strengthened (LTP), whereas in the second case will decrease (LTD). The right plot shows te relation between potentiation/depression and the inter spike time. As the time between two successive spikes increases, the change in the synaptic weight decreases.

2.4 Neuromorphic Hardware

As compared to the human brain, which only takes about 20 W of energy, computers consume a massive amount of power to simulate a neural network on general purpose hardware such as CPU or GPU. The latter especially became the standard to train and run neural networks taking advantage of the highly parallel architecture, which could significantly speed up computation. The use of GPUs in machine learning have unlocked the possibility to train deeper architectures with an increasingly high amount of parameters. Nevertheless, GPUs have a very high power consumption and are not specifically designed to simulate neural networks, being a general purpose graphic device. Even before the use of graphic

cards in research, scientists have developed Application Specific Integrated Circuits (ASIC)s to perform brain simulations [59–61], leading the way not only for more efficient brain and neural network simulations, but also a significantly different computer architecture which could overcome the Von Neumann bottleneck. In fact, in regular Von Neumann based architectures, the memory and processor are separated and the number of achievable computations is limited by the capacity of the communication bandwidth between them, whereas neuromorphic machines promise a significant reduction in bandwidth usage by providing a massive parallelization and memory/computation colocalization which would not require this continuous exchange of information.

In the following, some of the most remarkable hardware design of neuromorphic computing devices are listed:

- Neurogrid was one of the early successful neuromorphic chips that could simulate 1 million neurons with 1 billion synapses with 100,000 times less power consumption compared to a Blue Gene supercomputer [62]. The chip allows for simulations at different level of details by trading off the neuron model complexity and the network size, i.e. 1 million single compartment neurons or 100,000 ten compartment neurons. In other words, it can either simulate a lot of simple neurons or a few complex (multi-compartmental) neurons. The project is carried out by Kwabena Boahen's lab in Stanford University

- BrainScaleS is an European project started in early 2011 at the University of Heidelberg, Germany [63–66]. They have developed a device using wafer-scale integration, a manufacturing process that aims at reducing costs for defective chip replacement, by combining multiple chip modules on a single wafer allowing for the substitution of the specific defective module. How-

ever, in the BrainScaleS system, the wafer-scale integration is used to enable dense connectivity pattern, In fact, each of the wafer contains 48 modules, in turn containing 8 High Input Count Analog Neural Network (HICANN), comprising 128,000 synapses and 512 membrane patches. The connectivity of the individual neuron needs to be tuned depending on the network size. In other words, highly connected neurons can be simulated on smaller networks and vice-versa, i.e. 196,000 neurons with 256 synapses each (50 Million synapses in total) or 3000 neurons with 16,000 synapses each (48 Million synapses)

- Spinnaker [67] is a dedicated computational resource which acts to provide a digital platform to model spiking neural networks at large scale in real time. With an asynchronous, highly parallel architecture large numbers of small data packets can be processed, which in most applications represent voltage spikes being sent between neurons. This provides an ideal computational tool for event based processing. SpiNNaker has the added benefit of having a very low power design, dissipating only 0,1 W per core. If an entire 48 chip board is used with 17 active cores per chip only 8,16 W will be consumed which is far less than a High Performance Computer needed to process this scale of events in real time [68]. Furthermore, as a result of the sparsity of spikes the computational resource needed to process them will leave large portions of SpiNNaker idle during run time, allowing some cores to consume close to zero power. The chip allows for configurable neuron modelling as well for learning rules. Spinnaker is a scalable platform in terms of number of cores, ranging from 64 up to a million.

- IBM TrueNorth [69, 70] is a scalable neuromorphic device that aims at implementing fast and energy efficient computation on multiple and poten-

tially noisy sensory input. It is scalable in the sense that multiple boards could be stacked together to implement bigger networks. Each board contains 4096 cores with 256 neurons each and each neurons can have up to 256 synapses.

- Reconfigurable On-Line Learning Spiking (ROLLS) [71] is a mixed-signal analog/digital Very Large Scale Integration (VLSI) device for implementing on-line learning spiking neural network architectures with biophysically realistic neuromorphic circuits such as Short Term Potentiation (STP) synapses, Long Term Potentiation (LTP) synapses and low-power, low- mismatch adaptive I&F silicon neurons. The device comprises 128K analog synapse and 256 neuron circuits. The ROLLS neuromorphic processor can be used to carry out basic research in computational neuroscience and can be exploited for developing application solutions for practical tasks. In particular, its ability to carry out on-chip on-line learning allows for solving tasks that require the system to adapt to the changes in its input signals and in the environment it interacts with.

- DYNAP-SE2 is a device currently developed from aiCTX, a spin-off from the Institute of Neuroinformatics of University of Zurich and ETH Zurich. Similar to ROLLS it is a mixed analog-digital architecture. Each board features 1 k redesigned adaptive exponential integrate-and-fire analog ultra low-power spiking neurons and 65 k enhanced synapses with configurable delay, weight and short term plasticity. The asynchronous low latency communication infrastructure enables each neuron to communicate with up to 262 k surrounding neurons. The system is commercially available and currently applied in wearable devices for health monitoring or segmentation, tracking and recognition of objects in 3D, as claimed on the company's

website. However there is no thorough implementation details of the applications being unpublished work.

- Loihi [72] is a promising fully digital architecture developed by Intel with 131 k neurons and 130 M synapses. There are many boards released by Intel that differ in the number of chips being integrated ranging from 2 up to 768, allowing for implementation of large scale networks. Loihi can simulate LIF neurons and allows on-chip learning with programmable learning rules.

- The Google Tensor Processing Unit (TPU), unlike the neuromorphic hardware listed so far, which are specialized on the implementation of SNN, is an hardware accelerator for implementation of Neural Networks [73]. In particular, the processor is developed using a Complex Instruction Set Computer (CISC) style, as compared to the more popular Reduced Instruction Set Computer (RISC) design style. In RISC, only a very limited set of instructions are implemented at the lowest level of the architecture, whereas CISC also includes higher level operations, i.e. matrix multiplication, an ubiquitous operation in neural networks. In particular the TPU is a specialized hardware that speeds up operations and manipulation of Tensors, the main building blocks of neural networks. The TPU has been used by Google internally for few years and is now commercially available integrated in the Coral Development Board [74], a branch of the Google Research department. Very recently, also Asus has announced that it will integrate the TPU in their TinkerBoard computer [75], but price and availability are still unknown.

- The GeNN [76] library provides another interesting approach to simulate SNN. This software package, openly available, acts as an interface between the network model and the deployment on a regular GPU. Even though this

is not a way to deploy networks on a neuromorphic device, it is still a good way to easily test the network performance on more commonly available hardware in order to widen the amount of researchers running experiments on SNN. Additionally, [77] claims that GPUs outperform current neuromorphic solutions both in terms of speed and energy when simulating a highly-connected cortical model with GeNN.

Amongst all the neuromorphic devices discussed here, BrainScaleS, ROLLS and DYNAPSE use intrinsic low-level properties of the transistors to achieve very low latency and power consumption. In these hardwired processors, only the biases can be modified. Consequently, they support the implementation of only one (or very few) neuron models, where only the synaptic weights, time constants, neuron parameters and inter-neuron connectivities can be modified. On the contrary, processors like SpiNNaker and Loihi allowing to re-program the neuron dynamics in software to implement as many neuron models as needed. This enables greater flexibility at the cost of efficiency. BrainScaleS, SpiNNaker and Loihi also provide cloud-based platforms for remote use on their servers. However, while SpiNNaker and Loihi can be directly embedded in the robot for online signal processing, BrainScaleS can only be used for batch processing input data offline.

2.5 Datasets

To properly train a neural network in a supervised fashion, a big amount of annotated data is required. A number of datasets is available in the RGB domain such as ImageNet [78], MS COCO [79], or PASCAL VOC [80], just to name a few. To give an idea of the amount of data, ImageNet, which is the biggest

dataset available of the kind, contains a total number of images of 14 197 122, out of which 1 034 908 have bounding box annotations. Unfortunately, when it comes to event-driven data the availability of large datasets is much less. Most of the research papers that present machine learning techniques, also record their own dataset, which is typically small and constrained to the addressed task [15,38,41]. Recently, the research community has put some effort in making more datasets publicly available. The most interesting ones are listed below. Unless otherwise stated, all of the datasets are publicly available.

- In [81] the authors present an event based version of the famous MNIST dataset [53], which they call N-MNIST (Neuromorphic MNIST). The dataset contains recording from a DVS camera mounted on a motorized platform placed in front of a screen. The digits of the MNIST dataset are displayed on the screen while the camera performs saccadic motions in order to generate events. Using the same technique they also convert to spikes the Caltech-101 [82] dataset which contains 101 classes of objects;

- Poker-DVS [83], in which a card deck is rapidly browsed in front of the camera, as shown in Figure 2.6, and it is a useful benchmark for symbol recognition in fast conditions. The dataset has also been annotated with bounding boxes in [42] (not publicly available);

- For gesture recognition in [19] the authors present DvsGesture, a dataset containing 11 hand gesture categories from 29 subjects under 3 illumination conditions recorded with a DVS camera. Figure 2.7 shows a few snapshots of the gestures recorded in the dataset;

- NeuroConGD (not publicly available) is another gesture recognition dataset called which has been introduced in [84] with 17 different gestures performed by 4 subjects

- Multi Vehicle Stereo Event Camera (MVSEC), a dataset for depth and ego-motion estimation, is presented in [17]. In their work, a stereo setup with two DAVIS cameras is mounted together a LIDAR and IMU on several vehicles: a car, an hexacopter and a motorbike. The recordings are taken while driving around with these vehicles in different illumination conditions. They also provide ground truth in form of depth images and camera pose;

- DHP19 (Dynamic Human Pose), is a dataset for human pose estimation in [18], which contains 17 people performing 33 movements. Recordings contain both events taken from 4 DAVIS cameras surrounding the subject and ground truth is provided by a motion capture system;

- EV-IMO [85] is a dataset mainly focused on independent motion segmentation, but that could also be used for depth or ego-motion estimation. It consists of a total of 30 minutes recordings of an indoor scenario with moving objects. The room and the objects recorded are first 3D scanned, in order to generate automatic ground truth. In fact, during the recordings the pose of both camera and objects are taken using a motion capture system and the previously taken scans used to generate depth map and segmentation masks;

During this thesis work an annotated dataset for object recognition has been recorded by using the cameras mounted on the iCub robotic platform. Chapter 4 will provide further details on the content of the data and how it was recorded.

2.6 Object detection and recognition

This section will go more in the details of the tasks addressed in this thesis, that is object detection and recognition using event driven cameras. The detection part can be carried out using an attention mechanism and Section 2.6.1 analyses the

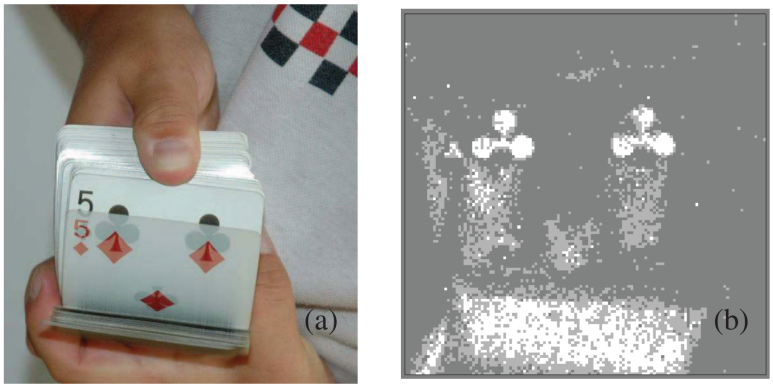


Figure 2.6: Example of data recorded in Poker-DVS [83] dataset

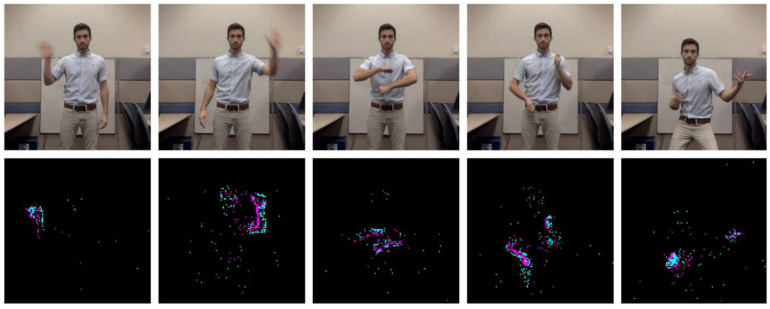


Figure 2.7: Example of gestures contained in the DvsGesture [19] dataset

methods available in literature, whereas Chapter 3 describes the implementation developed as part of this research. Section 2.6.2 provides information about the state of the art approaches to carry out object recognition and Chapter 4 and 5 will describe the proposed non-spiking and spiking implementations respectively.

2.6.1 Attention mechanisms for object detection

Attentional selection is crucial for biological organisms to react to the most important stimulus at any given time, like a fast-moving predator from which they must immediately escape, or a single red apple amongst green foliage. Fast selection is paramount to real-time interaction (and survival) of the system in a dynamic world and enables detailed further processing of a small region of the visual input, since all of the input cannot be fully processed by the brain in real time [86, 87].

Autonomous robots can similarly take advantage of attention mechanisms to reduce the computational load for visual processing when confronted with the vast amount of information in the world, and choose the most appropriate behaviour. As in all resource-restricted systems, in which it is impossible to fully process all sensor information simultaneously, choosing the most important sensory signals is fundamental. Attention solves this problem by identifying the most relevant regions of interest, and processing them sequentially in the order of decreasing relevance. This process is formalised in the concept of saliency map [88].

Most of the attention models developed so far follow the structure of [88],which introduces the idea of early representations of the visual scene, as a way to encode features of the stimulus, such as colour, orientation, intensity, disparity or direction of movement. An example of models that build on these principles can be found in [89–92], where several strategies to combine such low level representations in order to build up a saliency map are explored. In general the visual stimulus is preprocessed to extract the feature maps, i.e. using Gabor filters for oriented edges, Center-Surround filters for contrast detection, or even simpler operation to extract colour opponencies or light intensity [93]. Successively, neurons in each feature map spatially compete for salience, through long-range connections that extend far beyond the spatial range of the classical receptive field of each neuron. The maps are then merged together, so that the emerging saliency maps encodes for the most relevant stimulus regardless of which feature was responsible for it. The final saliency map finally goes through a Winner Take All competition, that selects the most salient location of the visual field. Eventually Inhibition Of Return (IOR) disengages the selection of the current winner, enabling an active scan of the scene, which subsequently selects different salient regions. The saliency computation merely depends on the stimulus appearance, selecting stimuli that are unique or different from the others. However [93] also

suggests that top-down biases could be used to modulate the feature map combination. These concepts are summarized in Figure 2.8, taken from [93].

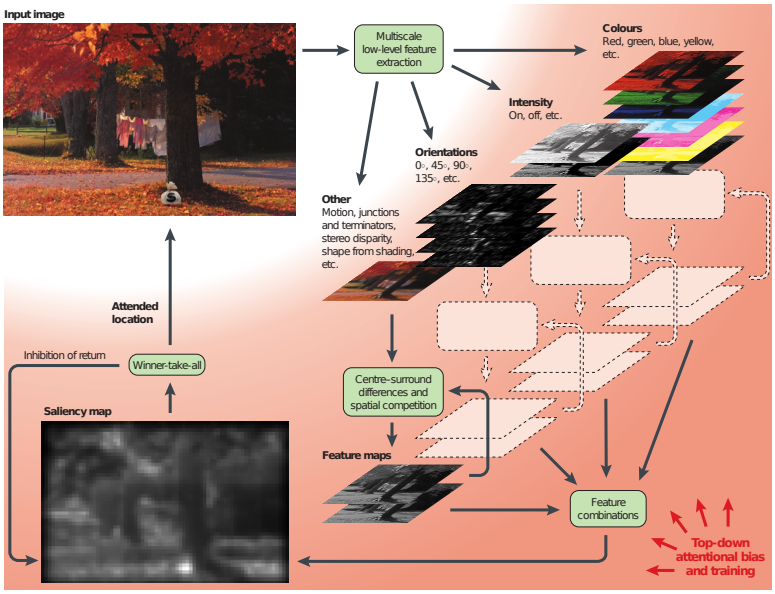


Figure 2.8: From [93]. Block diagram showing the typical model of attention system. The visual stimulus is preprocessed to extract the feature maps successively neurons in each feature map spatially compete for salience. The maps are then merged together to obtain a saliency map, which finally goes through a Winner Take All method, in order to get the most salient location of the visual field. Eventually, an IOR [94] approach could be utilized to enable an active scan of the scene. The model may also include top-down biases that modulate the way feature maps are combined.

Psychological studies have also discovered that attention in humans is not only driven by the features described so far, but also by their organization in the visual field [95]. In particular, we are mostly attracted by areas of the scene which may contain an object, even before we recognize what that object is. This knowledge has led to the introduction of the proto-objects concept [96]. Proto-objects are regions of the scene that *potentially* correspond to physical objects. This concept of "perceptual organisation" of visual scenes derives from the work of early Gestalt psychologists that developed "Gestalt laws", e.g. continuity, prox-

imity, and convexity [97]. The human brain is, therefore, more likely to focus his attention towards regions of the visual field that contain features potentially corresponding to objects. In [95] the authors demonstrate that the human attention is indeed mostly driven by such low level features more than top-down high level information, proving that computation of attention is carried out at a very early stage of the visual processing pipeline. In their experiments, they have shown that subjects presented with synthetic, non realistic representation of proto-objects have faster reaction time in performing a pretext task, in this case the recognition of a target's colour, as compared to the presentation of a very similar stimulus but without a proto-object present. The images used in the experiment are shown in Figure 2.9. The two stimuli do not contain any actual object, but only a set of shapes organized as if there is a (proto)object or not. Yet the subjects could achieve better results at performing the task when the target was contained inside the shapes that formed a perceptual organization typical of objects, proving that their attention was more likely to be captured by the presence of a proto-object.

To be behaviourally relevant in robotic applications, the computation of the saliency map and the selection of relevant stimuli have to be performed in the shortest possible time, while minimising the use of computational resources. In a previous study [98], it was demonstrated that using event-cameras [7] meets these targets, substantially reducing the latency and the computational cost of the feature-based saliency model initially proposed in [93]. In that model, the saliency of a stimulus is defined by features such as intensity, colour, orientation, etc.

However, recent work based on the concept of proto-objects proved to better

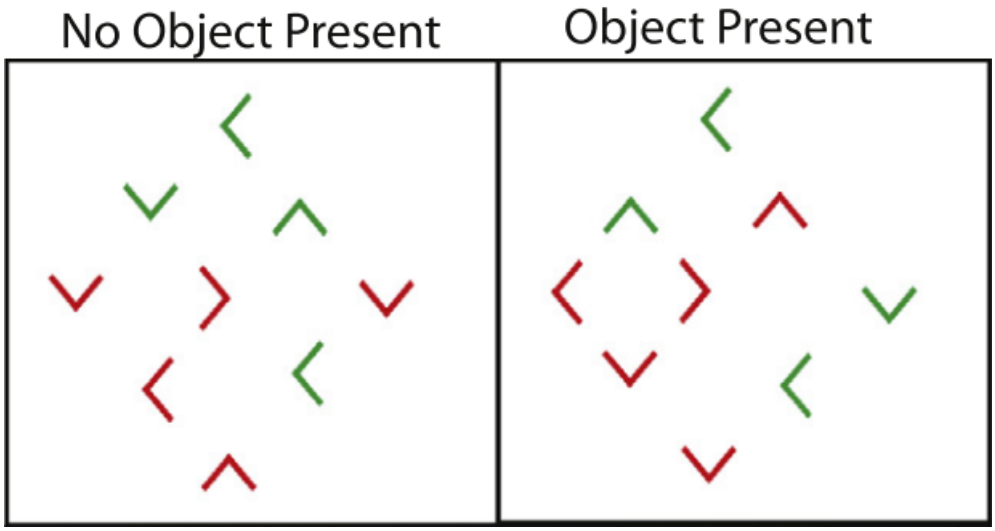


Figure 2.9: From [95]. Stimulus for psychological experiments on visual attention. Subjects were asked to tell the colour of a target that could appear inside or outside of the proto-object (enclosed region in the right image). Results show that the reaction time was faster when the target appeared inside the object proving that their attention was more likely to be captured by the presence of a proto-object.

explain perceptual saliency [3, 99, 100]. In particular Russell et. al. [3] implemented a subset of the Gestalt laws into a simple computational model. Such an approach is relevant for robots that operate within a human environment, as it is behaviourally relevant to, e.g., quickly locate potential objects upon which the robot can act. The algorithm [3] is divided into three main stages: 1. a centre-surround filter enhances the contrast of the stimulus and acts as pre-processing for the extraction of edges, 2. border ownership cells represent edges and, importantly, signal in their firing rates the location of foreground objects relative to the location of their receptive fields, as observed in primate extrastriate cortex [101, 102], and 3. grouping cells which respond to regions enclosed by several borders, thereby representing the presence of a proto-object [102]. Chapter 3 presents the implementation of a proto-object based saliency computation which uses the events as input.

2.6.2 Object recognition

The most successfully solved task that has brought deep learning to its current popularity is certainly object recognition [6]. Deep learning algorithms have reached amazing accuracy in recognizing objects in large datasets, and there is a big amount of work that tried to achieve the same results using event-driven cameras. Solving the recognition task with such sensors is more challenging as a result of the different organisation of the information. Additionally, the sensor resolution is typically much smaller than their frame-based counterpart, making it harder to recognize objects that appear smaller in the field of view. However, it has been demonstrated that the precise timing of events add valuable information to perform recognition. More precisely the fine temporal resolution provides up to 70% more information that conventional spikes generated from frame-based acquisition as used in standard artificial vision, thus drastically increasing the separability between classes of objects. [103]

Nonetheless, there are several works that have achieved remarkable performances that can be divided in two macro categories: spiking and not spiking. As described already in Section 2.2 there are many ways to treat the event stream and the nature of the AER allows for the use of spiking neural network [10] to perform object recognition task. These networks are harder to train due to their non differentiability, but allow for a significant reduction in terms of computational requirements and power consumption once deployed. One possible approach to use spiking architecture is to train a regular CNN with frames reconstructed by the event stream and then map the trained weights on a spiking architecture. To perform fast inference, the learned weights can be transferred to a SNN, which can be used to exploit the sparse event-data as spikes are individually propagated through the network, and are therefore well suited to event camera data.

This technique is presented in [37], in which card pips are recognized from a fast browsing deck with 92.5% accuracy. Similarly, [104] provides some guidelines to perform the conversion from ANN to SNN, by testing their algorithm on the MNIST dataset. However, the original dataset does not contain spikes, therefore it needs to be converted by generating a Poisson distribution of spike trains based on the pixel intensity value. To overcome this complicated training procedure, unsupervised approaches have also been used, as in [58], where they use the STDP learning rule. Again, the samples of MNIST are converted to spikes using a Gaussian model, then presented to the network, which has no previous information about the duration of the presentation or the moment in which a new sample will be presented to the network. This architecture can achieve a remarkable 96.58% accuracy, given the very limited assumptions being made.

However, the methods presented so far have some limitations. First the weight conversion from ANN to SNN leads to a drop in performance, as addressed in [45], and the networks trained on synthetic data, such as the Poisson generated data [105], do not typically yield good results when deployed on real sensory data [49]. Additionally, the MNIST dataset is a good benchmark as a starting point, but does not give information about how well an architecture would work with more complex inputs. To the best of my knowledge, there is no STDP based learning algorithm that can cope with more realistic scenarios.

[106] introduces HFirst, that is a spiking hierarchical architecture used for object recognition. The network is made of a layered structure of simple and complex cells [107] followed by a classifier and manages to achieve 97.5% accuracy on the same Poker-DVS [83] dataset, as well as another small dataset containing alphabet letters and numbers shown to a DVS with 84.9% accuracy. This

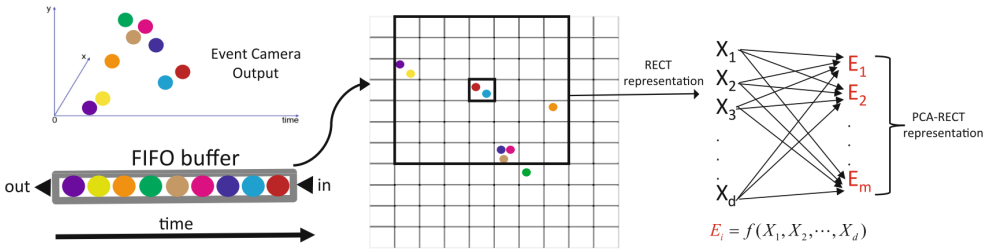


Figure 2.10: From [108]. Pipeline of the PCA-RECT representation. Events are filtered and stored in the RECT, then relevant features are extracted with PCA and sent forward for classification

architecture is interesting because the weights are trained directly on the spiking network, with a training algorithm that takes into account the time to first spike, exploiting the fine temporal resolution of event cameras. In addition it uses data taken from sensors, rather than synthetic. Section 2.3 also provides some hints on different strategies to train SNNs.

Non-spiking architectures can also be utilized for the purpose of object classification. To feed event data in a non-spiking architecture, as mentioned in Section 2.2, there are several ways to frame the spikes in a matrix, more similar to the RGB representation. In [38], they use a constant number of events to build the frame, because they need invariance with respect the speed of the object. In their work an architecture that combines Slow Feature Analysis [109] and Extreme Learning Machines [110] can achieve 1% classification error on a small dataset of 8 objects rotating in front of the camera. [42] shows good classification performances on a number of datasets adapting the famous You Only Look Once (YOLO) [111] architecture. Their network is fed with a matrix in which each element is connected to a pixel of the camera that they call leaky surface layer, which mimics the behaviour of a Leaky Integrate and Fire neurons. They also adapt the operations, such as convolution or pooling, making them asynchronous. They name the combination of leaky surface layer, YOLO architecture and event-driven

operations, YOLE. Not only they can perform classification but also detection with high accuracy on variants of MNIST and Poker-DVS. However, they show very poor performance on the more challenging dataset N-Caltech 101. In another work [108] the authors claim better performances on both N-MNIST and N-Caltech101 with 98.95% and 72.3% accuracy respectively. They achieve this result with an unconventional architecture that makes use of Principal Component Analysis (PCA) and a k-d tree search on the lower-dimensional feature space that PCA would generate to find a match with a set of candidate features that represent the classes to recognize. The input events are filtered according to their distance in space, such that events which are close in space and time are clustered together, and then collected in a matrix which counts the number of events clustered like this. The latter matrix, called RECT (Rectangular Event Context Transform) is sent through the PCA for dimensionality reduction and the resulting feature space (PCA-RECT) used for matching. This pipeline is illustrated in Figure 2.10. To the best of my knowledge this paper shows the highest classification performance on N-Caltech101. In an attempt to take the best from both frame and event-based sensors [112] introduces a transfer learning framework, explained also in Figure 2.11, named wormhole learning, in which detections from one sensor would compensate for failure of the other. In particular, they have taken recordings from both sensors mounted on the dash of a car driving both during day and night time. The idea is that event camera would work better than regular cameras in a low lighting situation, and vice versa. The wormhole learning process is divided in multiple steps: (i) a network is trained with a small portion of the daytime RGB data, (ii) the detections from the previous step are used to train a second network on events and (iii) the initial network retrained with detections from the event-based detector.

Deep learning has been applied to a variety of visual task in robotics, however

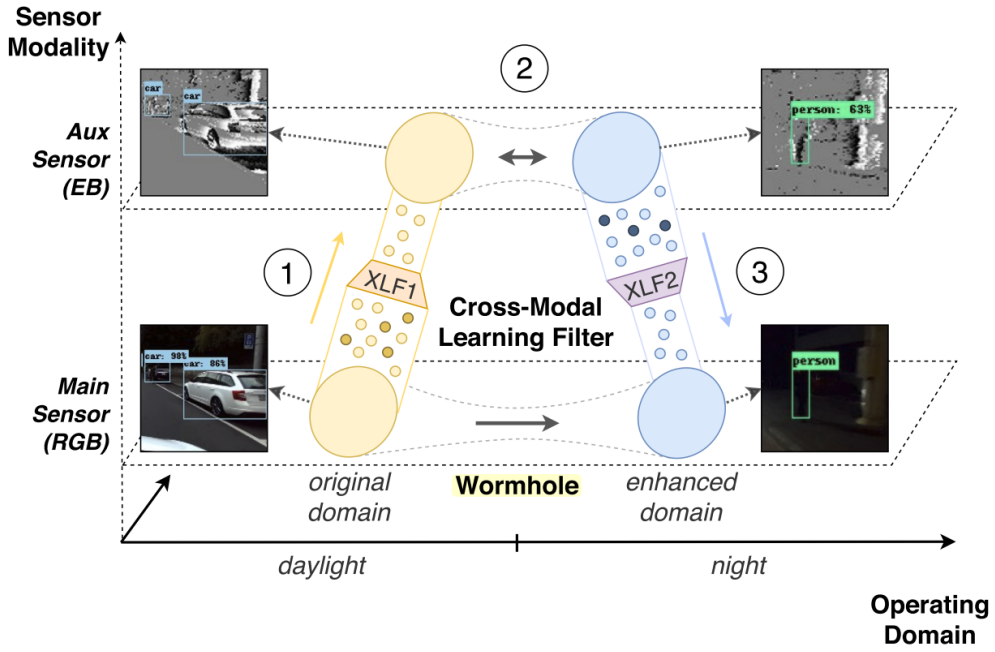


Figure 2.11: From [113]. The wormhole pipeline. Data from one domain are used to annotate and retrain the other domain in order to improve the performances.

deep CNNs are not without their limitations. In general, while some fundamental training can be done off-line, robots require the ability to learn new objects during their operation [114]. The computational hardware (e.g. GPUs) required to perform inference and further training imposes limitations on the autonomy of mobile robots, due to size and power consumption. In addition, (re)-training takes hours, (or days/weeks); reductions of the computational load will open more opportunities for deep-learning on autonomous robots.

In robotics, event-cameras have demonstrated a reduction in visual processing requirements for feature detection [115] and visual control [116], and shown fast tracking of objects [117] and Simultaneous Localization and Mapping (SLAM) [118]. The compressed visual signal – that corresponds only to changes in the scene – reduces the total amount of processing required, freeing (limited) resources for

other operations and reducing the overall power consumption. The high resolution temporal response with low latency gives robots potentially faster, more reactive, behaviours to visual stimulus. To also achieve competitive algorithm accuracy against the current state-of-the-art, deep-learning techniques need to be investigated.

Table 2.2 summarizes some of the techniques that have been presented in this section.

Paper	Recognition	Detection	Spiking	Data Representation	Dataset	Results	Remarks
[106]	x		x		Poker-DVS, Character	97.5% accuracy Poker-DVS, 84.9% accuracy Character	Layered structure of simple (Gabor) and complex cells followed by a classifier, Training by spike count, LIF with linear decay
[37]	x		x	Temporal window	Poker-DVS, Silhouette	92.5% accuracy Poker-DVS, 94.7% accuracy Silhouette	6-layers convolutional neural network trained then weights mapped to SNN wit same architecture
[42]	x	x	x	Leaky surface layer	Shifted MNIST-DVS, N-Caltech101	96.1% accuracy and 92% mAP on S-MNIST-DVS, 56.5% accuracy and 30.7% mAP on N-Caltech101	Adaptation of YOLO, NN components made asynchronous
[58]	x		x		MNIST	96.58% accuracy	SNN without any regularizing mechanism except for inter and intra map inhibition (WTA). Dual accumulator neurons allow to learn and infer simultaneously
[38]	x			Constant number of events	8 objects rotating	1% error	Combination of ELM and SFA
[108]	x	x		PCA-RECT	N-MNIST, N-Caltech101	98.95% accuracy on N-MNIST, 72.3% accuracy on N-Caltech101	Events are filtered according to their relative distance in space and time, then stored in a FIFO. A PCA is used to extract relevant features and a k-d tree search matches them to perform classification
[112]	x	x		Surface of active events [119]	Driving at day and night	58.05% mAP at daytime, 41.46% mAP at night-time	Transfer learning framework that takes advantage of benefits of both frame and event-based sensors.

Table 2.2: Overview of some of the most promising approaches for object recognition and detection

Chapter 3

Proto-object based saliency for event-driven cameras

The first problem being addressed is the object detection task. This chapter describes the adaptation of the proto-object-based saliency model from [3] to use the visual signal provided by an event-camera [9] mounted on the neuromorphic iCub humanoid robot [1, 2]. Results and methods described in this chapter have been presented in the IROS 2018 conference [5]. The attention mechanism allows the robot to quickly select object candidates and pass only the relevant regions of interest to further modules to perform the high-resolution processing required for recognition, grasping and manipulation. The algorithm is designed and tuned for the intended robotic application: an iCub robot quickly identifying potential objects. The event-driven algorithm is then validated on identical stimuli used in the original study, and also on new stimuli typical for the iCub environment.

The output of an event camera is a low-latency, asynchronous visual signal that typically describes the edges and contrast change of objects in motion. The proto-object approach to saliency, in which an object is defined by borders which are bound together at the grouping cell stage, is particularly suited for using

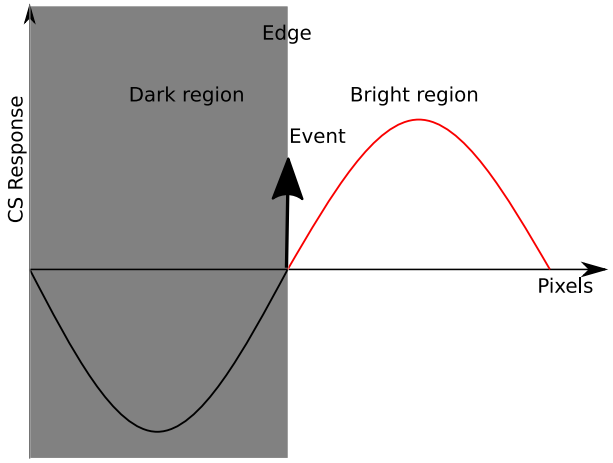


Figure 3.1: CS filters with ON (red) and OFF (black) centres respond to the light and dark side of the edge respectively. An event-camera will instead produce a an event directly on the edge location, with a polarity dependent on the edge's direction of motion.

event-camera signals, as it only responds to the edges and outlines of objects. However, as event-cameras perform a different computation at the silicon level, they produce a different output and visual representation than traditional frame-based systems. For this reason, it is necessary to adapt existing models, developed using input from frame-based cameras, to work with a new data representation and encoding. Interestingly, it was found that, as the event-camera more closely corresponds to a biological vision system, the processing layers in the original proto-object model designed to respond to contrast changes (i.e. centre surround cells [107]) are no longer required in the event-driven model.

3.1 Proto-object based saliency

The proto-object based saliency algorithm is formed by three different layers of processing: centre-surround, border ownership, and grouping cells.

The baseline algorithm, from which the event-driven version was formed, and

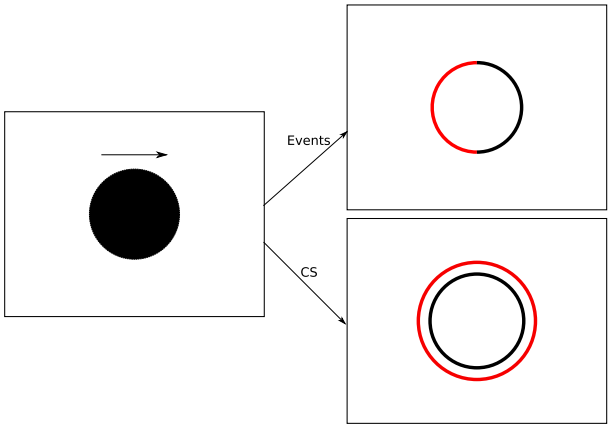


Figure 3.2: As an object moves, an event-camera produces events of opposite polarities on either side of the object. Processing an RGB image with CS filters instead results in negative (black) and positive (red) polarities on the inside and the outside of the object.

to which results are compared, is described in [3]. Attention models such as this have been inspired by [89] in which images are split into different feature maps, separately providing information about intensity, colour and orientation. The feature maps are then filtered by means of centre surround kernels, inspired by the organisation of visual cortex [120]. Final stages of the model aim to integrate and further process the feature maps to provide the final saliency map.

The main difference of [3], as compared to similar studies, is the exploitation of the proto-object concept [96], where (partially) closed contours that might correspond to objects provoke a strong activation on the saliency map. The final processing layers are divided in two main stages: border ownership and grouping. The first responds to individual edges that potentially form a (generally convex) border of an object. The second groups the elements of these potential borders, and regions enclosed by several object borders generate high response of grouping cells. Figure 3.3 shows the diagram of the original algorithm from which the event-driven implementation is derived. First, from the input RGB image a set of

feature map is extracted. In the case of [3], the maps encode for intensity, colour opponencies and orientation. The pyramid structure, depicted in Figure 3.3, is employed to respond to objects of multiple sizes. Each pyramid goes through the grouping mechanism where edges are extracted with CS filtering, then the borders are assigned a score depending on their probability of belonging to objects and finally, grouped together towards the object centre. The border ownership and grouping stage of the original algorithm are almost identical to this implementation and described in more details in Sections 3.1.3 and 3.1.4 respectively. Finally, the pyramid levels are merged together as well as the different feature maps.

As far as the Center Surround (CS) filtering is concerned there is a further analysis that needs to be done for the event-based implementation. The peak response of the initial CS filter is offset with respect to the edges that cause the cell's excitation. The result is that the filter response does not occur on the actual edges of objects. Figure 3.1 shows qualitatively this behaviour in the presence of a high contrast region, in relation to the spike (event) generated by the event camera. Figure 3.2 additionally illustrates the different locations at which positive and negative polarities occur. At the border-ownership stage, the output of CS cells is convolved with kernels generated from the Von Mises (VM) distribution. As shown below (Fig. 3.6), the filters have an offset which moves the saliency signal back to the edge location. Additionally the filter orientation has a preferred direction, eliciting a stronger response in the presence of edges whose convexity matches one of the filters. This is important for segmentation, because, according to Gestalt principles [97], objects tend to be convex around their centres. Border ownership cells are computed from the output of Complex cells, composed of odd and even Gabor filters, making responses to edges phase-

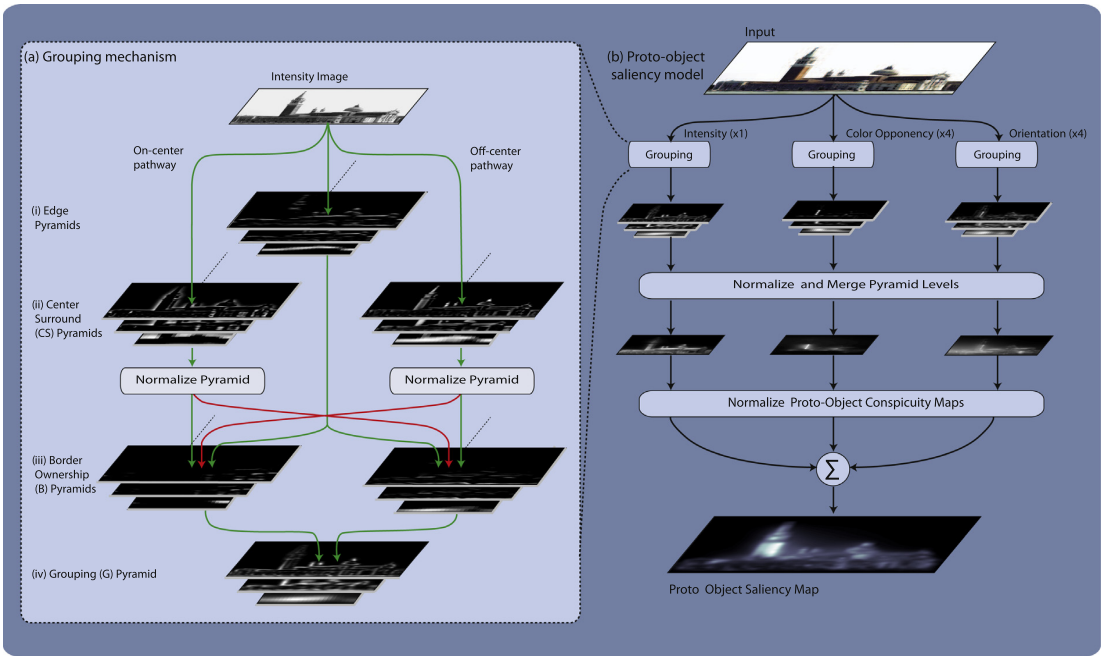


Figure 3.3: From [3]. This figure shows the diagram of the original algorithm from which the event-driven implementation is derived. First, from the input RGB image a set of feature map is extracted and organized in a pyramid structure to encode multiple scales. Each pyramid goes through the grouping mechanism where edges are extracted with CS filtering, then the borders are assigned a score depending on their probability of belonging to objects and then grouped together towards the object centre. Finally, pyramid levels are merged as well as the different feature maps.

invariant.

The response of border ownership cells is then integrated at the grouping cells stage, which is responsive to multiple borders that show proximity and continuity features [97]. The grouping cells also receive an inhibitory signal from filters with the opposing preferred side. This mutual inhibition suppresses the response from isolated edges, which are not particularly convex in either direction. Border ownership is a mutually exclusive property: a given edge either belongs to one object or another, and potential ambiguities are resolved in the human visual system in the form of perceptual rivalry [121], an effect that has been shown to

be implemented in primate visual cortex [122].

For more details on the original implementation the reader is directed to [3]; the present study adapts this algorithm to suit the visual signals of an event-camera.

3.1.1 Events representation

In this implementation, each event (or spike) v is represented by its coordinates, polarity and time stamp, $v(x, y, p, ts)$. The events received from the camera are accumulated into a binary matrix V as well as two other binary maps V_+ and V_-, encoding for positive and negative polarities respectively. All maps have the same size as the sensor, which in this case is 304×240. A fixed number of events in a time window W is considered at each iteration, but multiple events occurring at the same location are taken as a single one according to

$$V_+(x, y) = \begin{cases} 1, & \text{if } \exists\ v\ \in\ W \mid p_v(x, y) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

where $p_v(x, y)$ is the polarity of the event occurring at coordinates (x, y) that can be either 1 (positive) or 0 (negative). Similarly $V_-(x, y)$ is filled with events where $p_v(x, y) = 0$. The resulting maps are then sent to the border ownership module which selects the borders which more likely belong to objects.

As the event-camera does not have colour sensitivity, the colour-based feature map was not used in this implementation. In addition, fir simplicity, the orientation filter was also removed. The implication is that the algorithm will be equally responsive to edges, independent of the angle's distribution in the context of the scene. In comparison, an orientation filter map would instead make a single horizontal line more salient amongst many vertical lines. Such simplification was

made assuming that its effect on the saliency computation is minor. If necessary, an orientation filter can be reintroduced in future work.

3.1.2 Center-Surround

The pixels of event-driven cameras produce an asynchronous stream of events every time they detect an illumination change, making them natural contrast and edge detectors. Assuming a dark object moving on a light background, as shown in Fig. 3.2, processing frame-based camera inputs through CS filtering of opposite polarities would produce negative responses on the inside of the object and positive outside. The event-camera instead outputs "positive" (off-to-on) spikes on the leading edge of the object (the side towards which the object is moving) and negative spikes (on-to-off) at its trailing edge. Even though the information received from these two types of representations is very similar, the interpretation of the scene comes from different processes.

Furthermore, as reported in the original paper [3], Russell et al. used a center-surround mechanism of both polarities detecting a light object on a dark background and vice-versa, i.e., ON-center and OFF-center receptive field. However, as the events occur only on edges, implicitly containing the polarity information, it is possible to assume that the information that would normally come from the CS can be inherently found in the output of these sensors.

Not only exploiting this information can save some computation steps of the algorithm, but the spiking pixels would already provide the precise location of the edges in the image plane. It is therefore safe to remove the CS layer from the event-driven implementation. Figure 3.4 highlights the differences between the algorithm from [3] and the one proposed in this thesis.

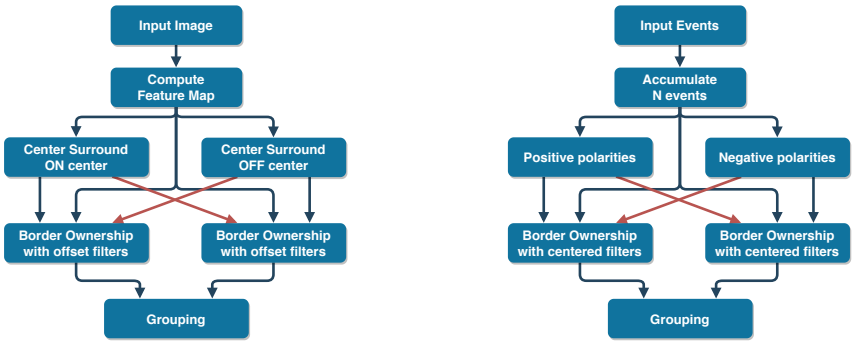


Figure 3.4: Comparison between border ownership algorithm in [3] (a) and ours (b). The red lines are inhibitory signals. In this implementation the Center Surround is not used as the polarities of the events already encode a similar information. Also feature maps are not computed and events are used as they come from the sensor. For clarity only one feature map is shown in (a).

3.1.3 Border Ownership

Despite using the same filters as in [3], modifications were made to tailor the model to the robotic application. In short, filters were made more rounded and less responsive to straight edges. The border ownership cells are modelled by the VM distribution:

$$VM_\theta(x,y) = \frac{\exp(\rho \cdot R_0 \cdot \cos(atan2(-Y,X) - \theta)}{I_0(\sqrt{X^2+Y^2-R_0})} \tag{3.2}$$

where X and Y are the kernel coordinates with origin in the centre of the filter, R_0 is the radius of the filter, θ its orientation and I_0 is the modified Bessel Function of the first kind. The ρ parameter was added to tune the arc length of active pixels in the kernel, allowing to change the filter's convexity. For values $\rho < 1$ the filter becomes more sensitive to convexities rather than straight lines, making it more suitable for the proto-object detection task. Proper tuning of this parameter improves the model robustness, making it more suitable to detect curved lines. It was empirically found that a value of $\rho = 0.2$ is good for detecting convexities

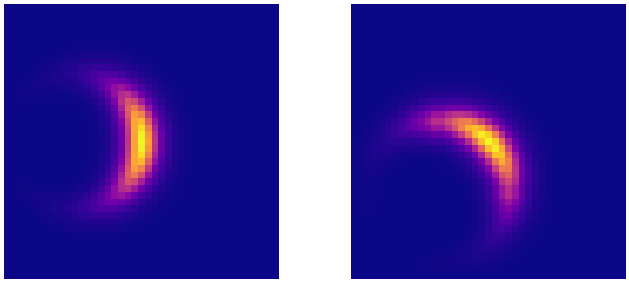


Figure 3.5: Example Von Mises (VM) filters used at the border ownership stage at 0 and 45 degrees. The centre of the filter is at the peak of the filter response, i.e. these filters are 'in-place'.

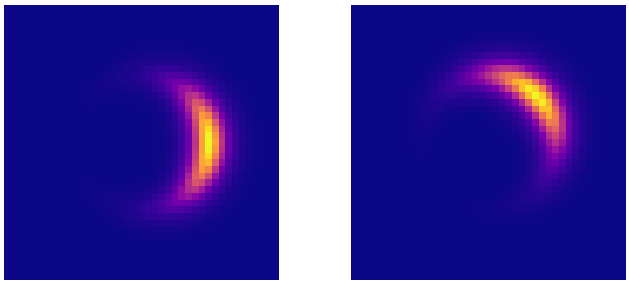


Figure 3.6: Example Von Mises (VM) filters used at grouping stage at 0 and 45 degrees. The centre of the filter is offset from the peak response.

while rejecting straight lines at the same time. The orientations used were $\theta = [0, 45, 90, 135]$. Additionally, since there is already an "in-place" response on edges due to the event-camera, as shown in Fig. 3.2, the filter response does not need to be moved back to the edge locations. At this stage filters centred on the position of the peak activity were therefore employed. To do so, a simple translation to eq. (3.2) can be applied, as follows:

$$\begin{aligned} X' &= X + R_0 \cos(\theta) \\ Y' &= Y + R_0 \sin(\theta) \end{aligned} \tag{3.3}$$

The resulting filters are shown in Fig. 3.5. The final border ownership response

is then computed as follows:

$$\begin{aligned} B1_\theta = V \odot (\lfloor V_+ * VM_\theta - wV_- * VM_{\theta+\pi} \rfloor \\ + \lfloor V_- * VM_\theta - wV_+ * VM_{\theta+\pi} \rfloor) \\ B2_\theta = V \odot (\lfloor V_+ * VM_{\theta+\pi} - wV_- * VM_\theta \rfloor \\ + \lfloor V_- * VM_{\theta+\pi} - wV_+ * VM_\theta \rfloor) \end{aligned} \tag{3.4}$$

where $\lfloor \cdot \rfloor$ is a linear rectification operation, $*$ a convolution, and $\odot$ an element-wise multiplication operator. The factor w weights the inhibition between competing polarities and orientations. The higher its value the harder it is for one orientation to dominate over its opposite, filtering out ambiguities. In other words, the border ownership cells are excited by the presence of a convex edge at a certain orientation θ and inhibited by activity of the same edge with opposite polarity and orientation. As a result, the borders who show only one preferred side and only one polarity are preserved, all the others get inhibited. This mechanism helps rejecting clutter and noise, because responses from both polarities are suppressed, as are straight lines. The rectification ensures that no inhibitory signal gets propagated to later stages of the computation. Finally, the element-wise multiplication by V masks the response of the border ownership located only where events exist. The result of the border ownership computation gives each event a score based on its likelihood of being a border regardless of its polarity, according to [101]. The resulting $B1_\theta$ and $B2_\theta$ encode the dominant orientations in the range $0 \leq \theta < \pi$ and $\pi \leq \theta < 2\pi$. The response of the border ownership layer is then passed to the grouping cells.

3.1.4 Grouping Cells

The activity of border ownership signal cells is grouped by the grouping cells (G). The grouping mechanism moves the energy towards the centre of the ob-

jects, from multiple boundaries, and thereby enhances proximity and continuity patterns [123]. Grouping is achieved using the same kernels as in Eq. 3.2, and the standard (non translated) filters are used, in which the centre of the filter does not coincide with the maximal response of that filter.

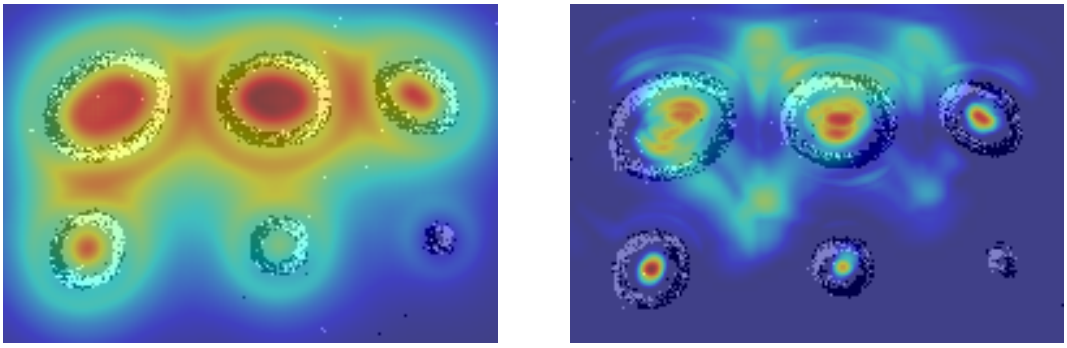
The excitatory component of the grouping cells is computed as:

$$\begin{aligned} G1 &= \sum_{\theta} B1_{\theta} * VM_{\theta} \\ G2 &= \sum_{\theta} B2_{\theta} * VM_{\theta+\pi} \end{aligned} \tag{3.5}$$

$B1$ and $B2$ are highly responsive to opposite convexity, this is why in Eq. 3.5 two opposite VM filters are applied. The aim of this is to move all the response coming from the object edges to the centre of the object. This process would also affect the object's surround, which can lead to ambiguity in-between objects. To counteract this effect, an inhibition mechanism is introduced reducing inter-object interference and preserving the saliency inside the object. First, the inhibitory signal is computed as shown in Eq. 3.6. Unlike Eq. 3.5, kernels of opposite orientation are employed, in the attempt of suppressing the activity on the non-preferred side of the border ownership cells:

$$\begin{aligned} G1^{*} &= \sum_{\theta} B1_{\theta} * VM_{\theta+\pi} \\ G2^{*} &= \sum_{\theta} B2_{\theta} * VM_{\theta} \end{aligned} \tag{3.6}$$

After that, the maximum value within the map gets subtracted from the remaining elements (Equation 3.7). The reason is that peaks of activity can be found inside the objects, where responses from all orientations overlap, and by subtracting the maximum value these peaks get suppressed. By taking the absolute value over the obtained map, it results in an inhibitory signal which



(a) *(b)*

Figure 3.7: Results of the calibration (a) without grouping inhibition and (b) with grouping inhibition. With correct inhibition calibration, high peaks of response can be found in the objects of sizes 20-30-40-50 pixels, which is the desired sensitivity for a typical humanoid robot application.

is mostly concentrated outside and in-between the objects. Fig. 3.7 shows the inhibition effects.

$$G1^* = |G1^* - max(G1^*)| \tag{3.7}$$

The final grouping is computed as follows:

$$G = (G1 - G1^*) + (G2 - G2^*) \tag{3.8}$$

3.1.5 Scale invariance

All the computation steps mentioned so far are performed at several different scales to obtain object size invariance/tolerance. To achieve this, the feature maps are arranged into a pyramid, in which each level is down-scaled by a $\sqrt{2}$ factor. To obtain the saliency map all the levels are collapsed into one by applying the normalisation method described in [89].

3.2 Validation and experimental results

The algorithm was implemented to work on the neuromorphic iCub humanoid robot [1], equipped with a pair of ATIS [9] sensors coupled with a pair of traditional cameras used for validation, both located in the robot's eyes. The two cameras share the same field of view and are calibrated to have pixel to pixel correspondence [5]. The ATIS camera has 304×240 pixels, whereas the frame-based camera has a nominal resolution of 1920×1080 but for this experiments the images are down-scaled to 320×240. The algorithm is implemented in C++ and runs online on the robot.

In the following experiments, the iCub looks at a number of either static or dynamic objects. Since static objects do not elicit any response from the event-cameras, the eyes are programmed to move in small circles introducing relative motion between them and the cameras. The circular motion has been chosen to span all possible orientations and capture all the edges in the scene.

3.2.1 Calibration

In a first set of preparatory tests, the parameters of the cells in the model are tuned in order to match the correct range of object sizes and positions that are relevant for the robot. Specifically, these are applications where the robot can grasp and manipulate objects, tailoring the model to optimally respond to objects whose size in the image plane is between 25 and 50 pixels. This range of sizes has been chosen according to the typical object that the robot can interact with, which is constrained by the robot workspace and grasping capabilities [124, 125]. A calibration step is carried out in a controlled scenario to set the right filter size for best response to the situation of interest. For this purpose, a calibration pattern with six circles of radii ranging from 10 to 60 mm is placed in front of the

cameras. In this condition, the pyramid depth is first set to 1, in order to find the smallest desired size by increasing the VM filter radius up to the point where the saliency shows a high peak in the middle of the object. Once calibrated for the smallest size, the number of pyramid levels can be increased until the algorithm responds to the largest desired object size. Results of the calibration process are shown in Fig. 3.7. With this empirical procedure, R_0 is set to 10 (see Equation 3.2) and the number of pyramid levels to 5. Table 3.1 shows the values used for each parameter of the model.

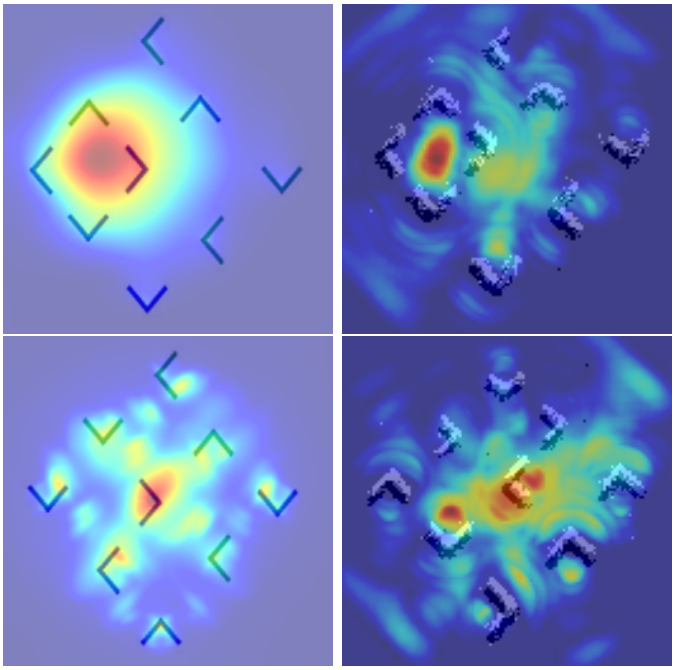
3.2.2 Comparison with the original algorithm

A series of experiments have been carried out in order to benchmark the model against the original work [3]. To this aim, a saliency map is first computed on a set of images using [3]. The same images have been then printed and showed to the event-driven cameras mounted on the robot.

Fig. 3.8 compares the response of both models to two pattern of corners. In the first pattern (top), four corners enclose a squared area; these are generally perceived as parts of the edge of an object, while the second pattern does not contain a similar "proto-object" [95, 127]. Both models correctly select the corners in the second pattern and show a peak activation of the saliency map in correspondence of the space enclosed by the four corners corresponding to the

Parameter	Value
R_0	10
ρ	0.2
ω	3
Pyramid levels	5
Orientations	0, 45, 90, 135

Table 3.1: Values of the parameters used in the experiments



(a) Frame-based camera *(b) event-camera*

Figure 3.8: The response to proto-objects for a closed stimulus (top) and to a similar stimulus but without the enclosed shape (bottom), comparing the original frame-based with the proposed event-driven implementation.

proto-object (left side of the image plane). In contrast, random orientation of corners (bottom) do not generate the perception of an object, and both models reflect this by generating weaker, disorganized activation patterns.

Fig. 3.9 compares the output of the two models to some images taken from the saliency benchmark dataset CAT2000 [126]. All Figures present a coherent and comparable response except for one, Figs 3.9c/3.9g. This might be due to the enhancement in curvature of the VM filter adopted that generates higher response in the presence of circular shapes like the one in the picture.

Fig. 3.10 shows the algorithm behaviour in a realistic scenario, with objects of different sizes and shapes as well as distractors (in form of cluttered cables)

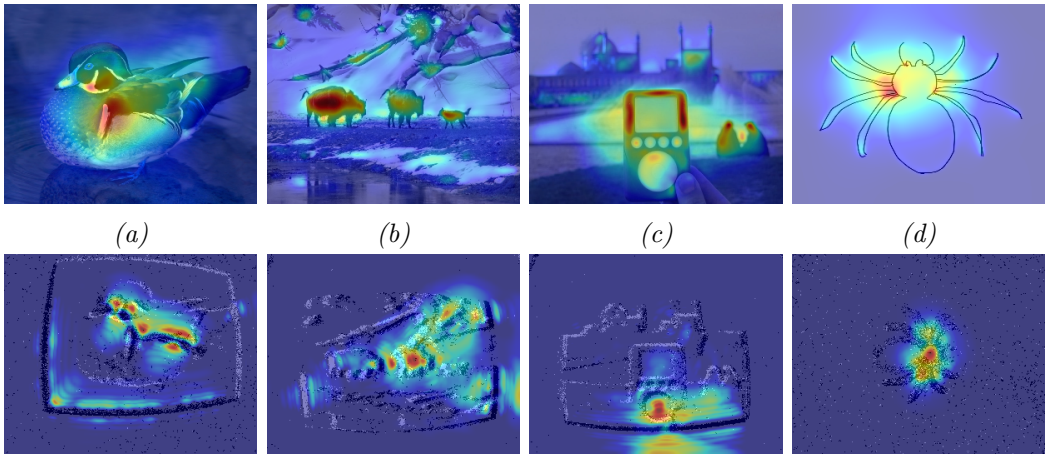


Figure 3.9: Comparison of saliency maps from the grouping cells response obtained with the original algorithm (first row) and this implementation (second row). The four pictures come from a dataset for saliency benchmarking [126].

placed in front of the robot. In this case, RGB images and events are recorded simultaneously [5], and the original model is applied on the RGB image. The output of both models is consistent, confirming that the adapted algorithm responds to the presence of proto-objects in the scene.

While the output of the proposed event-based and original models are qualitatively comparable, they produce different responses to the input images in Figures 3.10b and 3.10c, where the textured objects on the left side is less salient in the event-driven model. The strong inhibition factor $w = 3$ might explain this result. Moreover, the added inhibition mechanism explained in Section 3.1.4 would be useful to suppress the saliency of regions with events of both polarities, that often correspond to noise, clutter, or flicker stimuli.

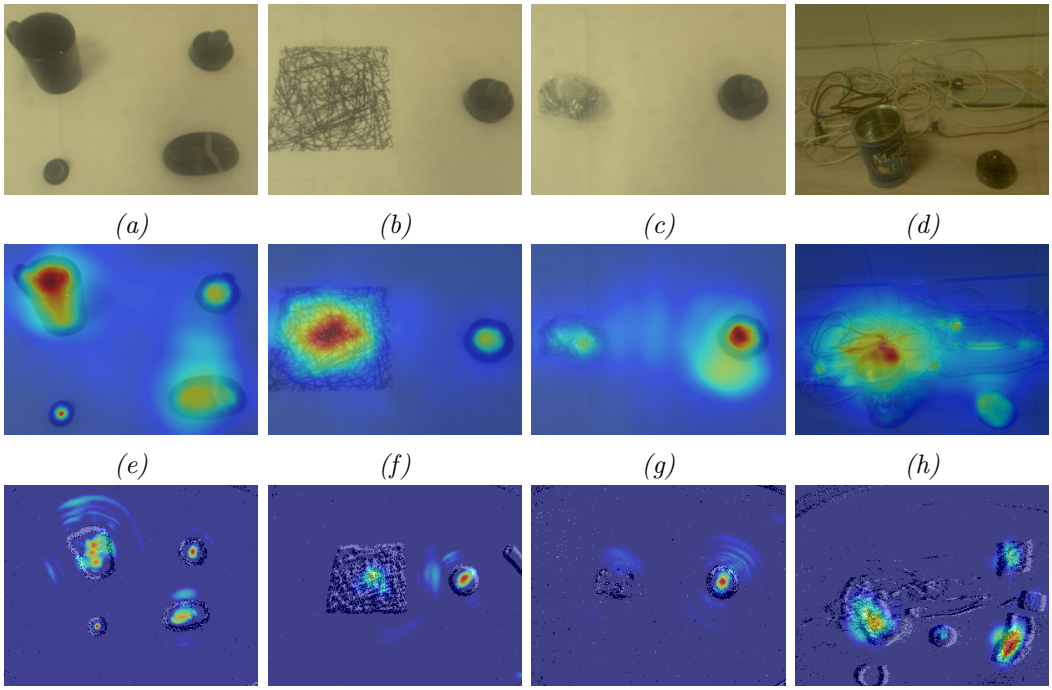


Figure 3.10: Saliency response for different stimuli including multiple objects, clutter texture and cluttered objects. The top row shows the original image, the middle row shows the response of the original frame-based algorithm and the bottom row shows the response of the proposed event-driven algorithm. The scenes do not appear as exactly identical because the event camera has a slightly larger field of view than the frame-based camera.

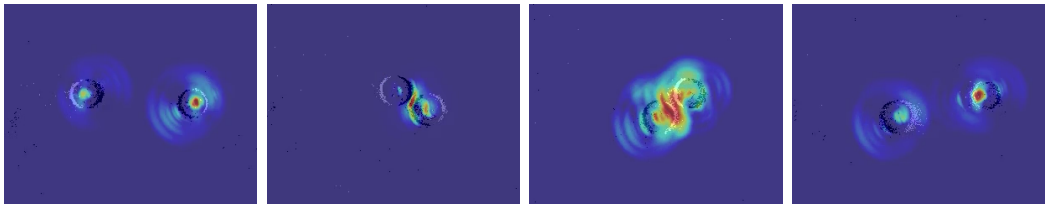
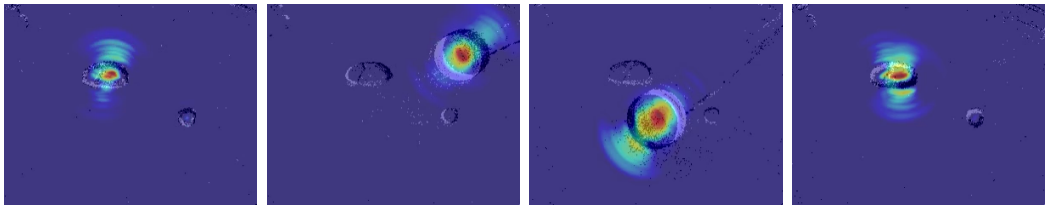


Figure 3.11: Effect of distance between objects. From left to right, the objects are first far apart (a), only when the objects share a contact point they are perceived as a single object (b,c). They are again detected as distinct ones with increasing distance (d).



(a) (b) (c) (d)

Figure 3.12: Saliency map for static and dynamic stimuli: (a) Two static objects are placed on the table and the focus of attention is localised on the left (bigger) object. (b, c) As the dynamic stimulus enters the field of view, it captures the robot's attention. When the moving object gets out of sight (d), attention goes back to the static object.

3.2.3 Moving objects

Finally, the event-driven model is tested with dynamic scenes. Fig. 3.11 shows some snapshots of a sequence where two objects roll on a desk and collide. The model shows two peaks when the objects are far apart. When the two objects are closer, the saliency map shows interference between the two objects, generating a single peak.

The role of inhibition in the grouping layer is crucial to suppress the activity elicited by edges with opposing curvatures, i.e. contours of two different objects which would increase saliency in the space between two objects. This mechanism works as long as the two objects are not too close: when two approaching objects are touching each other, the algorithm is not able to distinguish them anymore.

The grouping mechanism reduces the representation of objects to their simplest possible form. This is in agreement with the Gestalt law of proximity, where "objects or shapes that are close to one another appear to form groups". Intuitively, the collision between the two objects generates a large number of events which may cause the attention to get focused on that point.

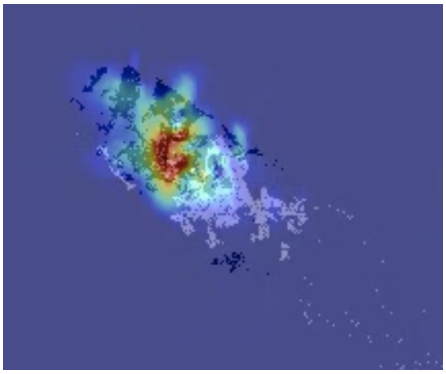


Figure 3.13: Saliency map of rapidly moving object

Fig. 3.12 shows snapshots of a sequence with the robot observing at static objects when a new object enters the field of view. In Fig. 3.12a, the static objects generate peaks in the saliency map. In Figs. 3.12b and 3.12c the dynamic stimulus captures the robot's attention as soon as it gets in the field of view of the camera. When the dynamic object leaves the field of view, the attention goes back on the previous object, Fig. 3.12d.

An advantage of the event camera is that it can process very fast motion because it is not limited by frame rates. To demonstrate that this advantage translates into saliency processing of rapidly moving objects, a ball was tossed in front of the cameras moving at a speed of ≈2000 px/s. Fig. 3.13 shows that this method successfully places attention on the ball.

3.3 Discussion

The aim of this work is to adapt a proto-object attention model [3] to work with the neuromorphic event-driven cameras embedded on the iCub humanoid platform. The overarching goal of this approach is to endow the robot with a

low-latency, computationally efficient attention system that is fundamental in an image processing pipeline for a robot which has to act in a dynamic and unconstrained environment. As event-driven cameras encode the visual signal in a radically different way with respect to frame-based cameras, it was necessary to tailor the model and correctly interpret the sensor output and the effect of the different filters on the novel input. Specifically, the event-driven sensor acts as edge extractor, functionally replacing the first layer of the frame-based model, based on CS filters. However, further modifications to the border ownership and grouping layers are required to correctly process the output of the event-driven camera. Specifically, ON and OFF events were separated in two parallel streams and the inhibition connectivity pattern in the grouping layer modified to take into account the difference between polarities in leading and trailing edges of objects.

We carried out preliminary qualitative experiments to prove the consistency of this implementation with the theoretical model and test its limits. In general, the implementation of the proposed model proposed produces an output that is consistent with the original model. The algorithm was tested both with static objects and images, and dynamic scenes, with moving objects. In the event-based representation, saliency is related to the speed of the object since increased speed increases the number of events of the faster object within the scene. Event cameras naturally produce bias towards this type of stimulus as the moving object generates the most events to the camera.

The C++ implementation [1] of the event-driven model runs online on the neuromorphic iCub and is capable of selecting objects in a range of sizes that are typically used when the robot performs grasping and manipulation tasks.

[1] `https://github.com/event-driven-robotics/bordercell-attention`

The use of the event-driven cameras – that efficiently compress the signal and performs part of the computation on chip – leads to a computationally efficient implementation. However, the proposed implementation is based on a hybrid solution, where the events are accumulated in frames that are then convolved with VM filters in the border ownership and grouping cells layers. While this implementation is certainly helpful in characterising the algorithm and proof that the results are comparable to the original model, a fully spiking implementation of the model will further increase efficiency and latency: Events elicited by the sensor travel asynchronously along the hierarchy, the computation is restricted only to the filters that receive input events and, as soon as there is enough activity in a region, the network can produce a result that can be almost simultaneous with the stimulus presentation. The spiking implementation of the proposed model and a quantitative analysis of its performance in terms of attentional selection, efficiency and latency are the goals of current development.

Chapter 4

Towards Event-driven Object Detection with Off-the-shelf Deep Learning

Last chapter has demonstrated how the event cameras can be used for the purpose of object detection, however it is not clear whether the object being attended could be recognized using solely the events. This chapter will provide an answer to this question, analysing the results of a frame-based classification algorithm when fed with events. Results and methods described in this chapter have been presented in the IROS 2019 conference [4].

Streaming camera images and applying complex processing algorithms is often computationally infeasible at a high frame-rate for standard autonomous robotic systems. One potential solution is to change the paradigm in which visual signals are coded, using *event cameras*. The sensors have great potential for use in dynamic robotic tasks, and this chapter will present work towards low-latency, low-computation, object detection and recognition, by first investigating the information content of the event-stream compared to standard frame-based cameras.

In particular, this work takes a step back from immediately attempting to exploit the event cameras for computational savings within neural networks and asks: does the signal of an event camera contain enough information to perform detection and recognition in typical domestic robot settings using today's state-of-the-art algorithms? It has been shown that the binary event output is enough to render high resolution grey-scale images if integrated over tens of seconds [128], but such a latency is infeasible for robotics. This work is instead inspired by [129], in which events are integrated only on the order of milliseconds and object recognition is still possible. In these experiments, the camera is stationary and only events from an easy to segment object are considered. Instead it is interesting to understand if similar results can be achieved in more realistic robotic scenarios in which the camera is mounted on a moving robot within a typical office environment. In this case, there is much more clutter, which introduces the problem of false detections. In particular the experiments performed will analyse the algorithm sensitivity to the choice of temporal integration for tasks of detection-only and detection-and-recognition.

Today's state-of-the-art in object recognition is performed using deep Convolutional Neural Network (CNN)trained on massive amounts of visual data [6]. CNNs take as input high-resolution, 3-colour-channel, dense image arrays, from which information is internally abstracted and classified. Even if recent work [130] has shown that the high resolution sampling increases the information content and separability between different objects with respect to the typical sampling rate of frame-based cameras, it is uncertain whether the binary data produced by an event camera contains the necessary information to differentiate objects from standard background clutter, which might have similar binary signatures. Given only high-contrast regions of objects (e.g. edges) generate a binary visual signal,

Figure 4.1: The ATIS event camera and the Python CMOS camera share the same field of view. The events (dots) are sparse and only occur at contrast changes on the edge of objects. The black region is that part of the field of view which is covered by the event camera, but not by the Python.

the colour and subtle texture changes that are used as cues in frame-based CNNs are not present. The visual difference between objects and parts of environment or people may not be sufficient to train object detection and recognition models. The non-linear differential mapping of the absolute visual intensity may not be sufficient to achieve subtle variation between pixels, which may impede the functional level of the convolutional networks.

The paradigm shift to a low-latency, high temporal-resolution, sparse information sensor gives the potential for reactive, low-computational-cost robotics applications. However, the variable temporal information is something that is not present in frame-based cameras and must be investigated to understand its effect on recognition performance. At the highest temporal resolution, a single event (a single pixel) does not provide any real information about the scene on its own, and accumulating events for too long can produce blur that masks fine

details, and warps the appearance of an object or scene. In addition, the optimal balance changes over time as the velocity of objects and/or the camera changes.

Therefore this chapter will present a pilot study to investigate whether event-driven data compression retains enough information for object detection and recognition, using state-of-the-art, off-the-shelf deep learning. As the initial focus is on investigating the accuracy, and not the computational speed-up, batches of events have been converted into frames which are fed into a SSD [131] CNN. The algorithm therefore runs at a limited frame-rate, but without modifying the network, to ensure that performance can be attributed solely to the data itself. Importantly, the robot and camera are themselves moving and background clutter is present in the event-stream, which can lead to misclassification if discriminatory information is *not* present in the data. Experiments aim to evaluate the sensitivity of the algorithms to the temporal parameters chosen. Such parameters affect a frame-based adoption (the integration window) or a SNN (typically the leaky-decay parameter). A good understanding of the effect of the temporal parameters will direct how training data should be collected and which models need to be trained for a more large-scale experiment. One of the mayor contribution of this work is the development of a method for bootstrapping frame-based methods to train and benchmark event-based models. This allows to cope with the lack of large annotated databases of event-based data.

4.1 Methods

This section describes the employed hardware set-up, which allows for automatic annotation of event-data, bootstrapping off mature frame-based algorithms, before describing the datasets used for training and testing the recognition models. An overview of the pipeline is shown in Figure 4.2.

4.1.1 iCub's Hybrid-camera

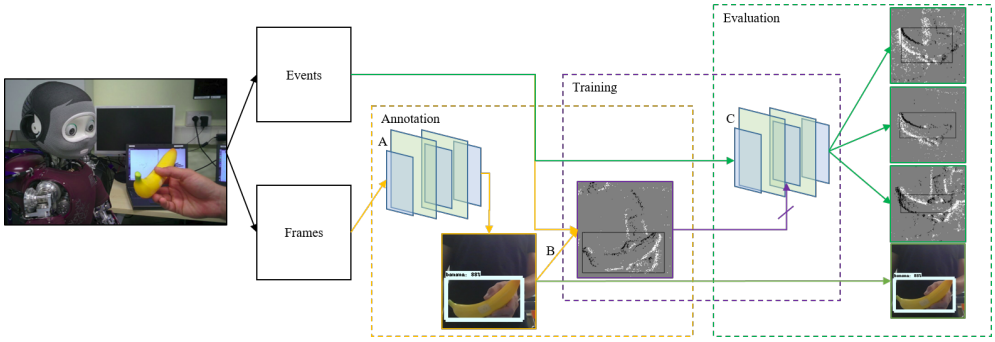


Figure 4.2: The proposed pipeline has three components: automatic event annotation bootstrapping from frame-based algorithms, model training with events using annotations as the learning signal, and evaluation comparing events of different window sizes and frame-based performance. The model (A) is the pre-trained Tensorflow Object Detection API [132], the operation (B) is the homography between frame and event cameras, and the re-trained model (C) is the deep-CNN [131]

The iCub humanoid robot used in this work is equipped with a hybrid camera set-up in each eye, comprised of both a frame-based camera and the ATIS event camera, which share the same lens, as depicted in Figure 4.3. The light is split to fall evenly on both cameras using refraction between the lens and the sensors themselves. Due to the way the cameras are mounted, the images are planar and share the same optical axis, however differences in camera resolution, physical size, and mechanical mounting imperfections still exist. A simple homography defining an affine transformation is all that is needed to translate and scale the event-pixel positions and re-project them to frame-pixel positions.

The homography, H_{vToRGB}, is calculated after visual signals are undistorted (both frame and events), using standard methods to estimate the intrinsic camera parameters using multiple observations of a known visual fiducial. A similar procedure can also be used to compute H_{vToRGB}, by feeding the points detected

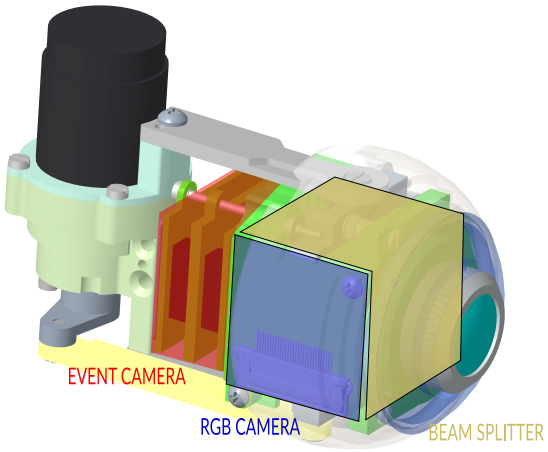


Figure 4.3: The hybrid camera setup mounted in the robot's eyes. The light comes through the pupil and half of it gets redirected towards the frame-based camera placed on one side of the eye, whereas the remaining light reaches the event camera placed on the back.

on the calibration pattern to the OpenCV method `findHomography`[1], which can estimate the affine transformation between a set of source and destination points.

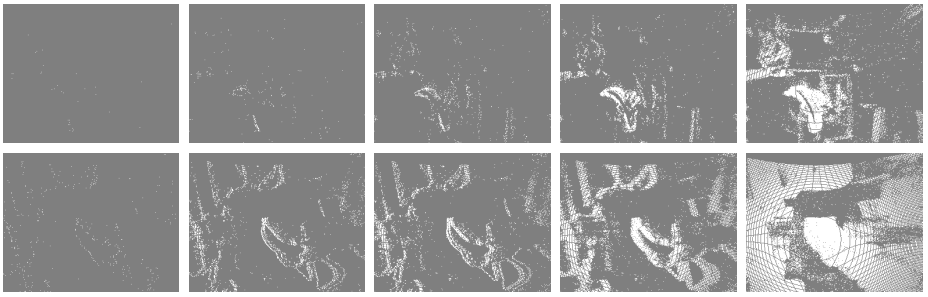
The result of the calibration is shown in Fig. 4.1. This hybrid-camera set-up is also useful to benchmark event-based algorithms against frame-based state-of-the art.

The iCub has a 6-DoF head that is controlled to direct the gaze of the robot to points of attention during operation. In the following experiments gaze version (left and right) and tilt (up and down) are employed to move the camera at known speed.

4.1.2 Automatic Annotation

The hybrid-camera system allows to bootstrap event-based learning algorithms by automatically annotating the data using off-the-shelf frame-based algorithms. Labelling of frame-based data is first performed using one of the pre-trained

[1] `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=findhomography`]



(a) 1 ms *(b) 10 ms* *(c) 30 ms* *(d) 100 ms* *(e) 500 ms*

Figure 4.4: Example training data (top row) and testing data (bottom row) for the banana with different temporal windows

architectures included in the Tensorflow Object Detection API [132]. Using the API it is possible to extract the location of objects in the scene and convert the resulting bounding boxes into their corresponding location in the event stream by applying the inverse of the homography H^{-1}_{vToRGB}. The result is an automatically labelled event-stream which can be used to train the event-based detection model using the annotations as the teaching signal.

The automatic annotation has the limitation that event-frames can only be produced at the frequency of frames (e.g. 30 Hz) and that false negatives cannot be learnt by the event-model. However, the former problem does not impact training, as it is performed off-line when frame-rate is not an issue. Although, using an event-driven approach to framing might produce more data more quickly. The effect of the latter problem is reduced during training by minimising motion blur in the frames and ensuring successful annotation for a wide variation in poses such that the event-model has an improved chance of generalisation.

4.1.3 Dataset Acquisition

4.1.3.1 Training

Datasets for training were captured by holding an object stationary in front of the iCub. The eye motors were then engaged to produce small circular motion

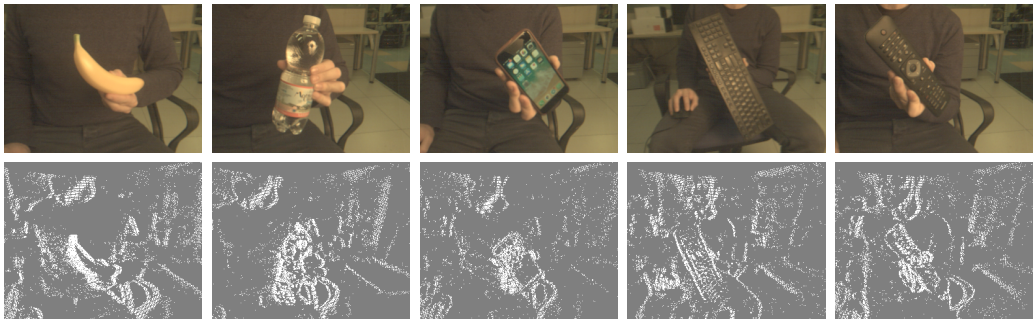
such that all object edges were at some point perpendicular to the eye direction of motion and given a full circle was performed, a full outline of the object was obtained. All possible directions were covered in 1 second of motion.

The object was slowly rotated and moved closer and farther to capture different object orientations and scales, for a total of 30 seconds. Each frame with a successfully classified position and object class was used to generate a corresponding frame of events and the annotated bounding box was used as the teaching signal. The process was repeated for 5 objects: banana, bottle, mobile-phone, keyboard, and remote-control, producing a total of 1125 images. This method was used to reduce motion blur in image frames to improve annotations, rather than due to a limitation of the event-camera.

All events within a temporal window were used to produce an *event-frame*. Event-frames were created with a pixel value of 127 for no event, and a value of 255 for the presence of an event. Alternative methods are described in Section 4.3. Different temporal window sizes are used as discussed in Section 4.1.4.

4.1.3.2 Testing

Datasets for evaluating the performance of the trained models were obtained by holding an object in front of the iCub cameras and moving the eyes left to right to produce a horizontal object displacement equal to the full image size. The speed of the eye was set to 30 degrees / s which corresponded to 150 pixels / s, or inversely, 6.7 ms / pixel, for each of the 5 objects. Event-frames were produced with different temporal windows, as in the training datasets. The testing dataset is different from the training dataset, as the latter generates motion in all possible directions. As event-cameras generate events only for changes due to motion of edges, in the testing dataset, edges parallel to the direction of motion do not generate events, making the dataset more difficult to extract objects.



(a) banana *(b) bottle* *(c) phone* *(d) k.board* *(e) remote*

Figure 4.5: Example data for each of the object classes. Events are depicted with a 30 ms temporal window

The ground-truth position and category of the object was annotated by hand, such as to evaluate the performance of both event-based and frame-based models, and to remove the dependence of events on correct frame annotations for the final evaluation.

4.1.4 Training a Deep-CNN with Event-data

Five models were re-trained for object detection and recognition from events. Each model was trained by using a different temporal window when producing event-frames; the window values were: 1 ms, 10 ms, 30 ms, 100 ms and 500 ms, and covered almost 3 orders of magnitude.

The model used was a SSD [131] object detection meta-architecture using the Inception-v2 architecture [133] as feature extractor.

As proved in other work [114], fine-tuning the network rather than training from scratch, allows to reuse features trained on big benchmark datasets, and specialize to the specific target task. However, in [114] the recognition task was performed on colour images, similar to the ones present in the original dataset used to pre-train the network. In the study case of this work the images generated from spikes have a very different appearance as opposed to the ones con-

Optimizer	RMSProp
Momentum	0.9
Batch size	24
Learning rate	0.004

Table 4.1: Training parameters

tained in any available image recognition dataset. Nonetheless, it was found that using the provided pre-trained models trained on the COCO dataset [79] (ssd_inception_v2_coco) was a valid starting point for training the model.

For training, horizontal flips and SSD random crop schemes are reused, as well as the data preprocessing schemes. All categories included in the recorded dataset were also part of COCO. The theoretical architecture of the automatic annotation network and the event-trained networks is the same.

Training was performed on a Nvidia GeForce GTX 1080 Ti GPU using the parameters shown in Table 4.1. All trainings were conducted by simultaneously evaluating the error on a test set containing roughly 30% of the total dataset, and iterating until the error reached its minimum or no-longer improving.

4.2 Results

The results obtained with traditional images are evaluated in comparison to frames generated from events. The aim of this study is to first check if there is enough information in the events to separate objects from people and background and then to detect the object class, also analysing the dependence from the temporal window used to generate the frames. Finally an experiment in low-light, and high-speed is presented, in order to validate and motivate the advantages of using an event-camera for object detection and recognition.

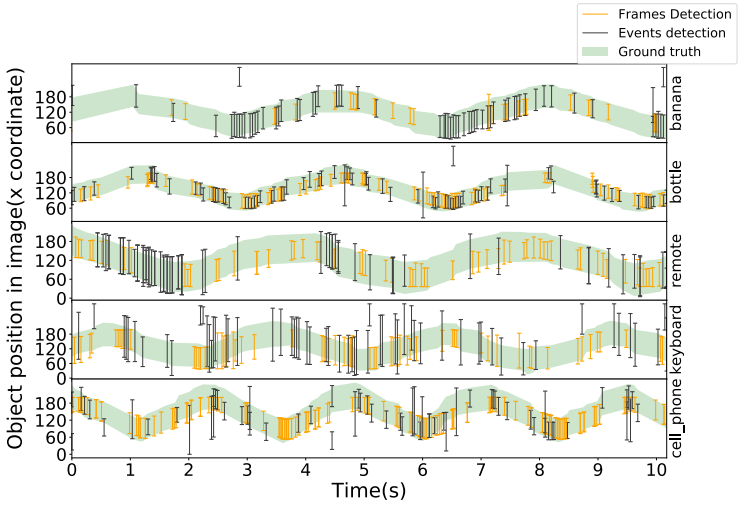


Figure 4.6: The detections over time for a 30 ms event temporal window, for each object. Since the objects move mainly horizontally, all the results are given against the x coordinate. The green shaded area denotes the ground truth location (i.e. within x_{min} and x_{max}), whereas each vertical bar represents a frame with a successful detection, where the top and bottom ends of the bars represent the detected location of the object.

4.2.1 Pipeline Output

The output of the trained model is the time and position of bounding boxes inferred directly from events. Fig. 4.6 shows a qualitative analysis of the event-based boxes and frame-based boxes in the X-axis only (as the eyes were moved only left to right, the Y-axis was constant) compared to the ground truth. For the temporal windows of 30 ms (both training and testing) the object localisation was comparable for both events and frames throughout the dataset, and for each of the different objects. This is an initial result that indicates that objects are separable from the background events using deep CNNs. Both events and frames overestimate the size of the keyboard, and the event-based detection model appears to perform slightly worse than frames on the remote and cell-phone categories.

4.2.2 Detection-only

As typically done for evaluating this task, detections were classified as correct if the Intersection Over Union (IOU) of the boxes (i.e. the amount of overlap) was above 50%. To evaluate the detection capabilities of the trained event-model the category classification is ignored and the average precision and recall is calculated over all objects (Fig. 4.7) depending on the temporal window used in the model training (x-axis) and the temporal window of the data used for testing (each series).

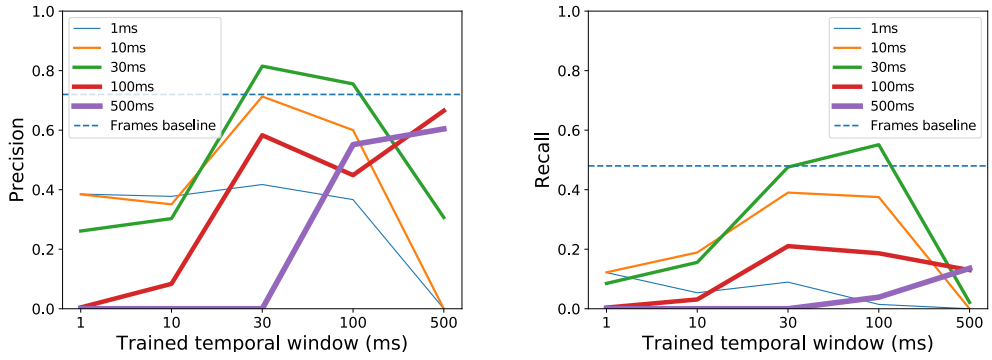
The 30 ms testing window performed with highest precision (80%) and recall (55%) when using the models trained at 30 ms and 100 ms (Fig. 4.7). This result is consistent with the example training and testing (Fig. 4.4) images, in which the edge thickness is comparable for these window sizes. The camera velocity was lower in the training datasets than the testing datasets leading to the smaller testing windows matching to the slightly larger training windows.

Both training and testing using 1 ms and 500 ms temporal windows had much lower recall values, which are again reflected by either under-representation of the object or blur from over-integration respectively (Fig 4.4). The 10 ms and 100 ms windows achieved comparable, but slightly worse, performance to the 30 ms windows.

The detection performance is therefore sensitive to the temporal parameters chosen, which must be selected based on the method of training, and the task in which recognition will be performed. However, and importantly, these results also provide a measure of the parameter tolerance which can be estimated at approximately 1 order of magnitude (in this case 10 ms to 100 ms).

The best window size for the datasets (30 ms) is comparable to the frame-based in both precision and recall. For this reason it can be concluded that the event-data does indeed carry the required information for object detection

amongst clutter.



(a) Precision *(b) Recall*

Figure 4.7: The precision and recall averaged over all objects for each training and testing temporal window

4.2.3 Detection-and-recognition

Recognition requires both the location of the object and the class of the object to be correct (IOU $> 50\%$). The event-based recognition models again perform comparably to the frame-based models, with the majority of recognition being correct, seen along the diagonal (Fig. 4.8). In the event data, the "bottle" and "remote" are often confused, while in the frame data the "phone" and the "remote" are often confused (see examples of these images in Fig 4.5).

4.2.4 Event-camera Advantages

The previous two experiments were performed in conditions in which both traditional and event-based cameras operate well. A final experiment was performed in low-light and with very-fast motion, in which a traditional camera is expected to fail, but a event-camera should theoretically still function. The motivation was to demonstrate why research into event-cameras is advantageous, as well as to validate the pipeline in difficult conditions.

	banana	bottle	remote	keyboard	cell_phone
banana	0.51	0.00	0.00	0.00	0.00
bottle	0.00	0.81	0.00	0.00	0.00
remote	0.00	0.33	0.15	0.00	0.00
keyboard	0.00	0.00	0.00	0.39	0.00
cell_phone	0.00	0.00	0.00	0.00	0.19

(a) Events

	banana	bottle	remote	keyboard	cell_phone
banana	0.12	0.00	0.00	0.00	0.00
bottle	0.00	0.76	0.00	0.00	0.00
remote	0.00	0.00	0.18	0.00	0.00
keyboard	0.00	0.00	0.00	0.47	0.00
cell_phone	0.00	0.00	0.39	0.00	0.50

(b) Frames

Figure 4.8: The recognition recall rate of each object using the 30 ms training and testing temporal window compared to the frame-based recognition

Figure 4.9 shows the comparison of the detection results using frames and events. The time is discretised along the x axis in the number of frames recorded during the experiment. The blue vertical bars represent the moment in time where either fast motion or low light condition starts (please notice that the fast motion state continues until the end of the experiment). Each cross represents a frame with a successful detection using either RGB images (blue crosses) or events (red crosses).

The experiment shows that the frame-based camera achieved many detections under standard conditions but failed under both fast motion and low-light. Alternatively, the detection rate of the event-based camera was unaffected by the fast motion, and still managed a functionally usable amount of detections even in the low-light conditions.

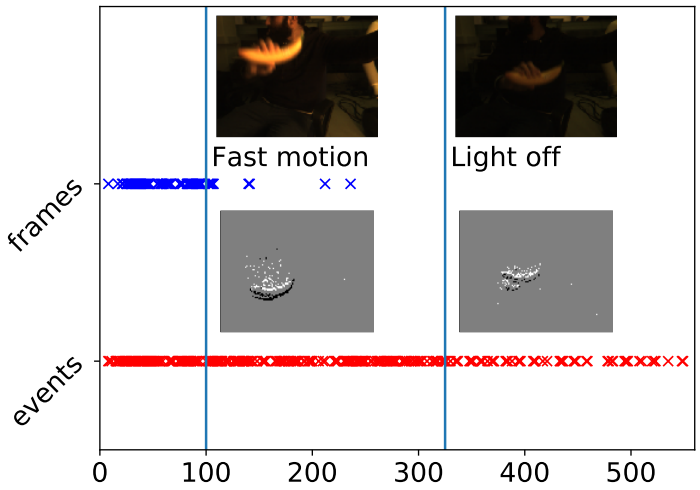


Figure 4.9: Detection and recognition is performed under extreme motion, and very-low light conditions, with comparison of the results using frames and events. The time is discretised along the x axis in the number of frames recorded during the experiment. The blue vertical bars represent the moment in time where either fast motion or low light condition starts (please notice that the fast motion state continues until the end of the experiment). Each cross represents a frame with a successful detection using either RGB images (blue crosses) or events (red crosses).

4.3 Discussion

The event-stream consists of additional information that was not evaluated in this pilot-study. Events have a polarity (whether the light increased or decreased), which was ignored. Pixel intensity values could also be set proportional to the event-count per pixel, or to the time since the last event occurred, to give a larger variation in the contrast with which edges are represented. Incorporating such information could lead to improved class separability but could also be irrelevant; for example, ignoring polarity might remove the need to train an object with both a dark and light background. Testing with more speeds and training data, as well as optimising parameters will also affect results. The presented study forms the

baseline to which further studies can be compared.

Ghosh *et. al.* [129] concluded that forming frames using a fixed number of events was superior to using a fixed time window. However, in a robotic context, in which the camera can be moving or stationary, or the depth and size of a moving object can vary widely, using a fixed number of events still results in large variation of the appearance of objects and background when framed. For this reason, the tolerance of the deep-learning architecture to the variation in the temporal window was investigated instead.

The number of objects is quite low compared to state-of-the-art algorithms that can manage thousands of objects. However, for testing the sensitivity to the temporal parameter choice, with only the limited number the effect presented itself. As the event-stream does not include colour and subtle textures, many objects can look similar even to a human (e.g. Fig. 4.5). Nevertheless, the deep CNN was still able to identify the unique features to distinguish between these categories proving that the method has some scalability. Again, pushing this limitations is the goal beyond this pilot-study.

One goal of this study was to identify best practices for gathering a much larger dataset for event-driven object recognition with deep learning techniques. For a larger dataset the training data will be made with faster and small eye movements to limit blur, while producing consistent event-frames across a wide variety of temporal windows. It should be possible to train on a single large window rather than multiple different windows. However, testing may need to be done on several temporal scales simultaneously if the object/camera speeds vary considerably, as the models tolerance to this parameter has its limits.

The algorithm recall rate is around 50% (Fig. 4.7) and looks sparse (Fig 4.6). However, frames occur at around 10 Hz and thus the frame-based approach is still producing 5 detections a second, which is usable for practical purposes. Even

still, it is possible to push events to a much higher rate, however currently events are only tested at the same rate as the frames to simplify the comparison.

Chapter 5

Object recognition with a Spiking Neural Network

So far it was proved that events can be used for the purpose of both detection and recognition, however, the classification system used in 4 does not take into account the different data format and, as a consequence, does not take advantage of all the information contained in the events, such as the fine temporal details. This chapter will take a step further by implementing an architecture that is specifically designed to process event data. This work was carried out in collaboration with professor Emre Neftci using a Spiking Neural Network (SNN). As described in Section 2.3, training a SNN may be problematic due to non-differentiability of spikes and therefore the impossibility of direct application of gradient-based learning such as Back Propagation (BP). Nevertheless, there is an increasing body of works focusing on how to overcome some of the constraints given by Back Propagation (BP) [52, 134–138] and professor Neftci has also been focusing on how to exploit the latest algorithms in order to train a SNN [51, 55, 139]. These works aim at going beyond the assumptions needed by BP, which are not compatible with a neuromorphic setup, both on a biological and practical point of view.

In the traditional training pipeline of NNs, first the forward pass computes neurons activity and generates an output, which is compared with a target in order to estimate a loss function. The resulting error signal is then propagated backwards via synaptic connections that are symmetric to the ones used in the forward pass. The network parameters can thus be adjusted by taking the derivative of each synaptic weight with respect to the error via recursive application of the chain rule [140]. Generally, the error signal is propagated with high accuracy i.e. as a double precision floating number. Therefore BP works under the following assumptions:

1. symmetric synaptic weights between back and forward pass, also known as the weight transport problem [138]
2. use of multiplication with derivatives and activation functions
3. use of high precision error signals
4. alternation of forward and backward passes

Such assumptions lead to a number of problems that limit the algorithm scalability. Before performing an update, the input must be propagated all the way to the last layer and then back, increasing the training time as the network gets deeper, and also delaying the update of intermediate nodes which need to be on idle, waiting for the downstream gradients to become available. This is known as the update locking problem. The latter also results in the use of backpropagated error signals which are non-local to the weight being updated, requiring the network to maintain its state for the entire duration of the forward pass, significantly increasing the memory requirements. Additionally, the use of high precision floating point parameters and complicated operations, increases even further the computational requirements of the BP algorithm.

In a recent work [52], a remarkable discovery has paved the way for a new training procedure which does not require the strong assumption of symmetric synaptic connections. In what they call Feedback Alignment (FA) algorithm, the weights used to propagate the error backwards are replaced by fixed random values, showing not only that there is no need for such constrained connectivity, but that any random projection of the error signal to the previous layers would suffice for training the network. The term Feedback Alignment comes from the observation that, over training time, the forward weights tend to align to the random ones. Figure 5.1 shows how the angle between the two weights matrices shrinks over training. The reason behind this happens is still unclear, however intuitively, it can be supposed that there is a flow of information that pushes the two weights to align and once this happens the backward pathway starts to contain meaningful information that can push the update along the right direction. Despite the lack of understanding, such behaviour is consistent with the non-symmetric connectivity in biology. There is no evidence that the neurons need such assumptions in order to learn and moreover, they do not need a global knowledge of other synapses weights. In fact all of the learning that takes place in the brain is performed locally at the neuron level and FA shows that this could be modelled into a neural network in order to develop more efficient and scalable networks.

However, FA still utilizes information which is not local, delaying the parameter update until the error signal becomes available. To address this problem, [56] introduces the first multi-layer architecture that updates the weights in a supervised fashion only using information which is locally available at the synapse level. To achieve this, each layer of the network gets connected to an intermediate output classifier with auxiliary targets via random fixed weights, similar to FA.

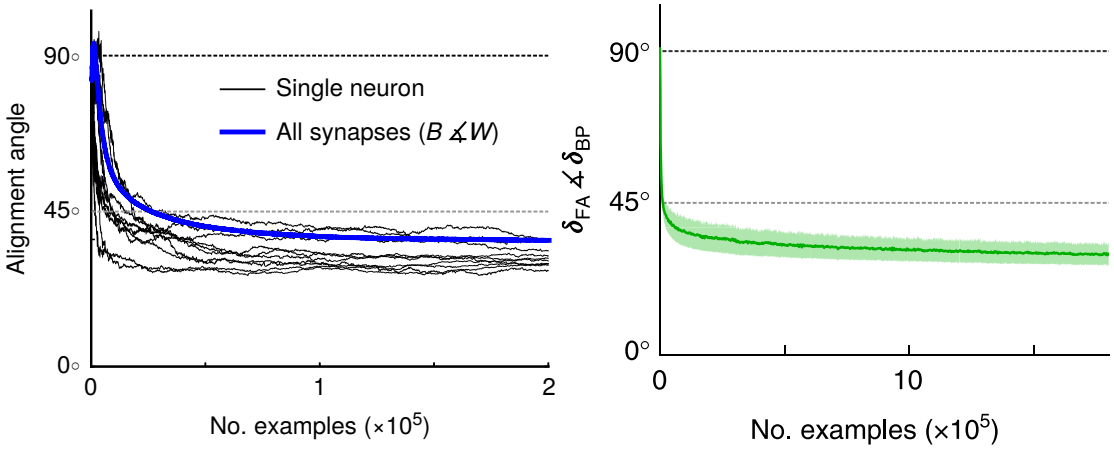


Figure 5.1: Figure from [52]. The two plots show (a) how the weights W tend to align to the random ones B and, as a consequence, (b) how the direction of the update driven by FA gets closer to the one driven by BP.

In this way, the update can be performed locally both in space, by using variables which are available at the neuron level, and in time, by doing the update as soon as the signal passes through the layer and without having to wait for the last layer to have computed its output. Remarkably this approach has outperformed FA and at the same time has relaxed even more the assumptions of BP, making one further step towards a more biologically plausible and scalable learning algorithm.

Building on these results, [51] has developed the Event Driven Random Back-propagation (eRBP) algorithm, that aims at delivering a spike-driven learning rule which at the same time overcomes the weight transport and the non-locality problem. The main core of the algorithm is a three-factor learning rule driven not only by pre and post-synaptic spikes, but also by an error signal, as introduced in [137]. The latter paper argues that this learning rule could be applied to either supervised, unsupervised and reinforcement learning paradigms, depending on the nature of the error signal, an idea that was also reported in [52]. In the case of eRBP, the error is the random projection of a loss function computed locally

at each layer of the architecture, similar to [56]. The eRBP dynamics for synapse j of neuron i can be summarized as follows

$$\Delta w_{ij} = T_i \; \Theta(I_i) \; S_j^{pre} \tag{5.1}$$

where T_i is the random projection of the error, S_j^{pre} the spike train of presynaptic neuron j, $\Theta(I_i)$ is the derivative of the spiking neuron's activation function evaluated at the total synaptic input I_i. For a spiking input, the derivative computation would result in a non-smooth function which is hard to train on and is thus replaced by a simple approximation to a boxcar function:

$$\Theta(I_i) = \begin{cases} 1 & \text{if } b_{min} < I_i < b_{max} \\ 0 & \text{otherwise} \end{cases}$$

The relaxation of the gradient computation is useful for the purpose of numerical optimization, but it does not affect the forward pass, which would still contain the hard threshold of the spiking neuron model [55].

All of these concepts are embedded in the Deep Continuous Local Learning (DeCoLLe) architecture [20], a spiking network with locally trainable layers, no symmetric synaptic weights and trained with a three-factor learning rule. Figure 5.2 shows a block diagram of a possible network configuration. More details about DeCoLLe will be provided in the following sections.

5.1 The neuron dynamics

The DeCoLLe implementation utilizes a LIF neuron model. The driving parameters of the model are the traces P and Q which describe the state of the membrane and the current-based synapses respectively. Each spike from an afferent synapse

causes an injection of current into the postsynaptic neuron which results in an increase of its membrane potential U, which sums all the input activities coming from presynaptic neurons. In the following the post and presynaptic neurons will be referred to with the subscript i and j respectively.

$$U_i(t) = \sum_j w_{ij} P_j(t) + R_i(t) \tag{5.2}$$

The term $\sum_j w_{ij} P_j(t)$ represents the Post Synaptic Potential (PSP), that is the immediate increase that occurs whenever a new spike is emitted, weighted by the synaptic weight of the synapse connecting neuron j to neuron i. The PSP is driven by the input current, which in turn depends on the spikes being emitted on that particular synapse. The reset term $R_i(t)$ sets the membrane potential back to its resting value after the neuron i spikes.
In turn, the dynamic of $P_j(t)$ is defined as:

$$\frac{\mathrm{d}P_j(t)}{\mathrm{d}t} = -\frac{P_j(t)}{\tau_{pot}} + Q_j(t) \tag{5.3}$$

where τ_{pot} is a temporal decay factor which causes $P(t)$ to naturally decay to 0 in absence of input. The dynamic of $Q(t)$ instead is driven by the incoming spikes from presynaptic neurons:

$$\frac{\mathrm{d}Q_j(t)}{\mathrm{d}t} = -\frac{Q_j(t)}{\tau_{syn}} + S_j(t) \tag{5.4}$$

Again there is a decay factor τ_{syn} that pushes $Q(t)$ back to 0 when no spike is received. The spike term $S_j(t)$ is set to 1 whenever the membrane potential of neuron j exceeds a certain threshold θ, resulting in:

$$S_j(t) = \Theta(U_j(t) - \theta) \tag{5.5}$$

where $\Theta(\cdot)$ is the Heaviside unit step function. For the sake of simplicity θ can be set to 0.

Finally, the dynamic of the reset term $R(t)$ is described by the following equation:

$$\frac{\mathrm{d}R_i(t)}{\mathrm{d}t} = -\frac{R_i(t)}{\tau_{rp}} - w_{rp}S_i(t) \tag{5.6}$$

where w_{rp} is a tunable parameter that regulates the magnitude of the reset, τ_{rp} the length of the refractory period, and $S_i(t)$ the output spike. The reset only occurs right after the neuron spikes i.e. $S_i(t) \neq 0$, and gradually goes back to 0 allowing the neuron to fire again.

This formulation is in agreement with the Spike Response Model (SRM) which models also the response to the neuron's own spike [141].

From the equations above it is possible to derive the formulation in the discrete domain obtaining:

$$U_i[t+1] = \sum_j w_{ij}P_j[t] + R_i[t] \tag{5.7}$$

$$P_j[t+1] = \alpha P_j[t] + Q_j[t] \tag{5.8}$$

$$Q_j[t+1] = \beta Q_j[t] + S_j[t] \tag{5.9}$$

$$S_j[t] = \Theta(U_j[t]) \tag{5.10}$$

$$R_i[t+1] = \gamma R_i[t] - w_{rp}S_i[t] \tag{5.11}$$

where $\alpha = \frac{\Delta T}{\tau_{syn}}$, $\beta = \frac{\Delta t}{\tau_{pot}}$ and $\beta = \frac{\Delta t}{\tau_{rp}}$ are the decay factors that regulate the leak in $P[t]$, $Q[t]$ and $R[t]$ respectively. Using this formulation it is possible to implement the SNN behaviour as a special case of a RNN, in which the cur-

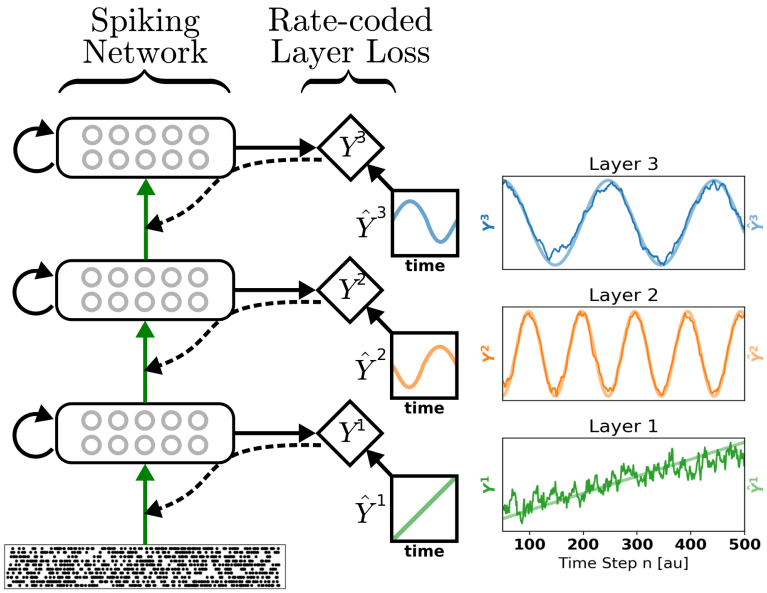


Figure 5.2: Figure from [20]. The DeCoLLe architecture. Each layer is independent from the others and can be trained to output arbitrary targets (Y^1, Y^2, Y^3). Training happens locally via random projection of the error signal (black dashed line). The green arrows represent the connections with trainable weights. Finally the state of the neurons is carried forward in time via recurrent connections (black curved arrows).

rent state of the network depends both on the input and on the previous state. Such dependency is visualized in Figure 5.2 as self connections between the layers.

5.2 Derivation of the learning rule

Given the formulation provided in Section 5.1, the network of spiking neurons can be considered as a special case of a RNN. This means that it is possible to utilize learning algorithms that have been applied to RNN. In this case an approximation of the Real Time Recurrent Learning (RTRL) [142] is implemented. The latter has the advantage that it does not require a precisely defined training

interval, however, it requires non-local communication. Deep Continuous Local Learning (DeCoLLe), instead constraints the variables being used to compute the gradient to be local to the layer of neurons being trained.

The learning rule can be derived starting from the definition of the derivative of the loss function with respect to the weights:

$$\frac{\partial L_l}{\partial W_l} \tag{5.12}$$

where subscript l denotes the layer where the gradient is being computed. In order to enforce the locality of the gradient propagation, the derivative are only computed between variables at layer l. To compute the loss function L_l locally, an auxiliary target is provided at each layer. The target could be arbitrary, but it was found that the use of the same final layer target yields good results, because each layer tries to compute the best features to perform the classification, and all the successive layers will build on the previous features to improve them [56].

By applying the chain rule it is possible to rewrite 5.12 as:

$$\frac{\partial L_l}{\partial W_l} = \frac{\partial L_l}{\partial S_l}\frac{\partial S_l}{\partial U_l}\frac{\partial U_l}{\partial W_l} \tag{5.13}$$

In Equation 5.13 the term $\frac{\partial L_l}{\partial S_l}$ denotes how a change in the spikes would affect the loss. In other words, it can be interpreted as an error signal e_l to be propagated through layer l.

Equation 5.5 provides the formulation of the spiking term as $S_l = \Theta(U_l)$, therefore, the second right hand side term of Equation 5.13 can be rewritten as:

$$\frac{\partial S_l}{\partial U_l} = \frac{\partial \Theta(U_l)}{\partial U_l} \tag{5.14}$$

Solution of this equation is problematic because it involves the differentiation of a step function which is zero everywhere except for zero where it is undefined. To overcome this problem a surrogate gradient is utilized in order to relax the non-smooth spiking non-linearity for purposes of numerical optimization [55]. The term is therefore replaced by a boxcar function:

$$\frac{\partial \Theta(U_l)}{\partial U_l} \simeq \sigma'(U_l[t]) = \begin{cases} 1 & \text{if } -b \leq U_l \leq b \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

where $\sigma'(\cdot)$ denotes the surrogate gradient function and b delimits the boxcar boundaries. In this implementation $b = 0.5$.

The last term left to compute is $\frac{\partial U_l}{\partial W_l}$. U_l can be replaced with its definition provided in Equation 5.2 yielding:

$$\frac{\partial U_l}{\partial W_l} = P_l + \frac{\partial R_l}{\partial W_l} \tag{5.16}$$

In the latter it appears a dependency on the reset term R_l, which is hard to compute because it requires backtracing the neurons' history. Furthermore, it mainly serves the purpose of a regularizing term, preventing the neurons from firing at high rate, and does not add any meaningful information for learning. Its calculation can therefore be dropped and its regularizing function compensated by manually add a regularization directly in the loss as explained in Section 5.3.

The final three-factor learning rule is therefore given by:

$$\Delta W_l[t] = \rho \underbrace{e_l[t]}_{error signal} \underbrace{\sigma'(U_l[t])}_{postsynaptic} \underbrace{P_l[t]}_{presynaptic} \tag{5.17}$$

with ρ being the learning rate. The weight update depends both on pre and

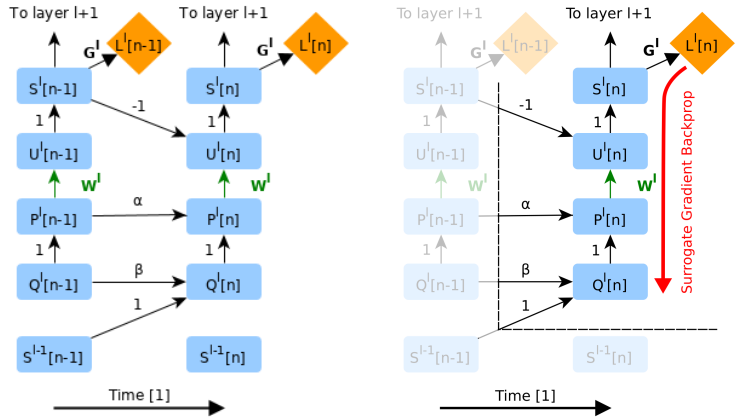


Figure 5.3: Figure from [20]. Deep Continuous Local Learning (DeCoLLe) computational graph unfolded in time. The gradient can be computed as soon as L_l is available and is only propagated locally. G_l is a linear random classifier which does not get updated, so the only trainable part is W_l

postsynaptic activity as well as the error signal. All the variables necessary for the computation of $\Delta W_l[t]$ are readily available within the layer (local in space) at the time the forward pass is being computed (local in time). Figure 5.3 shows the computational graph of the DeCoLLe architecture unfolded in time. The gradient can be computed as soon as L_l is available and is only propagated locally. By construction the availability of L_l is instantaneous because it depends only on timestep t. In the figure, G_l denotes a linear random classifier which does not get updated, whereas the trainable part is indicated by W_l.

One of the main advantages of DeCoLLe is that the gradient can be computed using the automatic differentiation tools already available in modern machine learning frameworks. This not only allows the portability and reproducibility of the code, but also the exploitation of optimized training routines which take advantage of GPU parallelization to accelerate learning and inference. Nonetheless, the network being inherently spiking, the trained parameters could directly be

mapped in neuromorphic hardware for deployment. As opposed to [37] or other similar approaches where the weights of an ANN gets mapped to SNN, here the training happens already in a network that takes into account the spiking dynamics, hence it would not be necessary to perform any weight conversion whatsoever. Additionally, as depicted in Equation 5.17, the learning rule is relatively simple to compute as it only involves comparison and additions, it is plausible to implement the learning directly on chip [143].

5.3 Methods

Previous sections provided a detailed mathematical formulation of the neuron dynamics in both forward and backward passes that shows how the layers are completely independent from one another. In fact, when designing the network it is possible to stack as many layers as the task at hand may require. Additional computation could also be interposed in between two successive DeCoLLe layers in order to perform i.e. spatial pooling or dropout. Another choice regarding the network's architectural design is the neuron connectivity between layers. This is defined specifically by the connection between P_l and U_l, which determines how the information coming from presynaptic neurons gets interpreted and propagated forward. For instance, layers could be connected with convolutional or linear patterns depending on the type of data being analysed. The implemented architecture has been optimized performing grid search and is illustrated in Table 5.1. Convolutional layers were used to compute features to cope with the visual data, and the feature size progressively reduced in order to extract more informative features in deeper layers.

The dataset used to validate the network is an expanded version of the one described in Section 4.1.3, with a total of 11 classes of objects and 4 different

Layer	# Out Features	Feature Size
ATIS	2	304×240
Downsample	2	32×32
7×7 Conv	128	32×32
Pooling	128	16×16
Dropout 0.5	128	16×16
Fully connected	11	1
7×7 Conv	128	16×16
Dropout 0.5	128	16×16
Fully connected	11	1
7×7 Conv	64	16×16
Pooling	64	8×8
Dropout 0.5	128	16×16
Fully connected	11	1

Table 5.1: Network's architecture. The input from the camera is downsampled and split into two features encoding the two polarities. The rest of the network is composed by three layers with decreasing number of parameters. The output of each fully connected layer represent the intermediate readout on which it is possible to compute the loss and perform the local weight update as explained in Section 5.2

instances per class. The network receives as input two event images containing positive and negative polarities respectively. Each sample provided to the network is a $500ms$ long recording of an object which is sliced in 500 chunks of data with duration of $1ms$. Each slice is represented as a binary matrix filled with zeros everywhere except for the locations where events have occurred during the short temporal window. It is safe to assume that the number of events that occur during a millisecond is low enough to well approximate the asynchronicity of the SNN. Furthermore, within such a short period of time it is nearly impossible to gather enough information to produce a meaningful classification, proving that the network is indeed properly maintaining a memory of the recent history, even though past neuron activity is not explicitly stored.

As loss function Mean Squared Error (MSE) was employed, plus a regularizing

term that penalizes high firing rates:

$$L_l = \frac{1}{2}(Y_l - \hat{Y})^2 + \lambda 1_l \langle ReLU(U_l + 0.01)\rangle + \lambda 2_l \langle ReLU(0.1 - sig(U_l))\rangle \quad (5.18)$$

where Y_l is the output of the classifier of layer l and $\hat{Y}$ the expected output, which in this case is the one-hot vector with 1 in correspondence of the correct class of object to be recognized and 0 elsewhere. Subscript l is omitted in $\hat{Y}$ because the same target is used across all layers. In the regularization part the $\langle\cdot\rangle$ is a mean operator and $sig(\cdot)$ denotes a sigmoid, which approximates the step function. The two regularizing terms force the membrane potential to stay below the threshold and keep the firing rate low. Such sparsity is especially important when carrying out the computation asynchronously, i.e. on dedicated hardware, to keep communication bandwidth and power consumption requirements as low as possible.

5.4 Results

The performance of the network is evaluated in terms of classification accuracy on the test set. The dataset is randomly split in training and testing set containing 70% and 30% of the samples respectively. The accuracy is computed separately at each layer and plotted against the training epochs, together with its standard deviation (shaded area) as shown in Figure 5.4. The accuracy increases with the depth of the network, but that the performance increment becomes less pronounced after layer 2, proving that the network has already learnt good features at that stage. Overall the network reaches an accuracy of about 0.6450, 0.8502, 0.8683 at layer 1, 2 and 3 respectively.

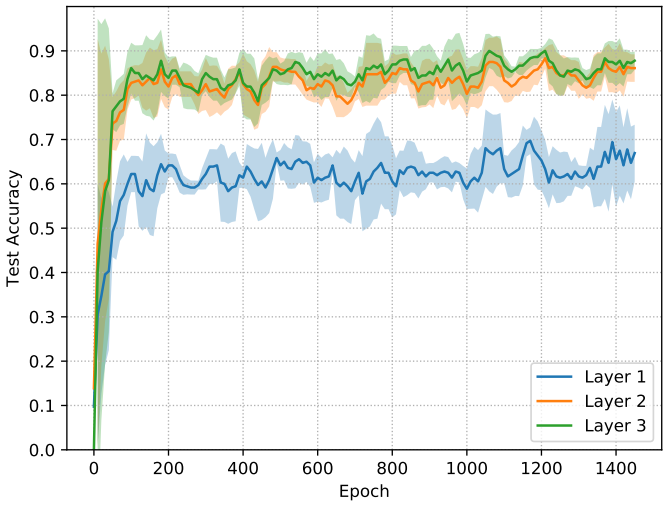


Figure 5.4: Test accuracy of DeCoLLe network over training epochs. Accuracy is computed every 10 epochs locally at each layer and plotted with its standard deviation (shaded area). As the network grows deeper the performance increases, but the increment becomes less pronounced.

To track the quality of the regularization the activity rate of the network was analysed, that is the percentage of neurons that spike at each time step in average. Figure 5.5 shows how the average of active neurons drops over training, proving that the regularization term is actually taking effect. The activity rate for each layer is 0.0986, 0.0301 and 0.0418, which means that most of the neurons are silent at each time step, resulting in computational requirements saving and power consumption decrease when deploying on a neuromorphic platform.

5.5 Discussion

This chapter presents the implementation of a biologically plausible, local learning rule, that allows training of a SNN exploiting available machine learning software tool for GPU parallelization. The neuron dynamic is mathematically formalized in order to treat the SNN as a special case of an RNN, allowing the

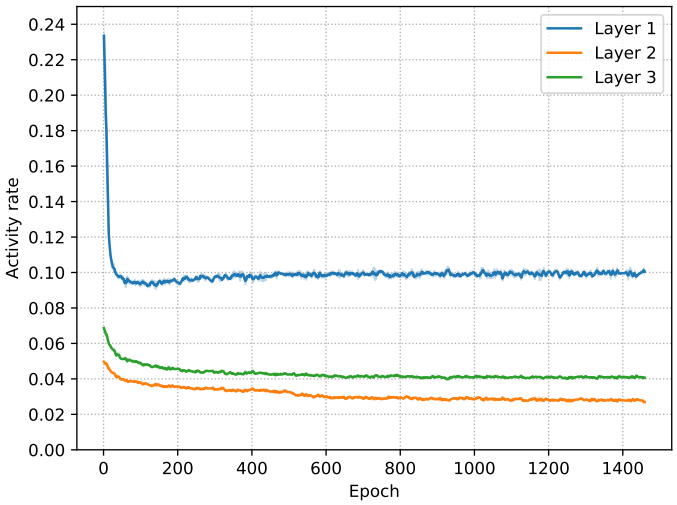


Figure 5.5: Mean activity rate of the network expressed as percentage of spiking neurons at each time step. The regularization forces this number to be slow in order to enforce sparsity, resulting in a low number of spikes with big savings in terms of computational requirements and power consumption.

use of learning procedures of recurrent networks. Building on this assumption the DeCoLLe learning rule is derived, overcoming the assumptions made by BP. The network has been trained to perform object recognition on recordings taken from the iCub robot in presence of background clutter generated by the camera ego-motion, achieving classification accuracy about 87%.

This result opens the possibility of a robotic implementation in order to provide a low-latency, low power consumption, asynchronous classification system which can also deal with fast dynamics or poor illumination condition, given the capabilities of the sensor. DeCoLLe takes full advantage of the data nature, potentially working at any time-scale. Given that no explicit history of the neuron is retained, the network is already showing a good adaptability to time scale. Still, the temporal parameters τ_{pot}, τ_{syn} and τ_{rp} require manual tuning but other

approaches can be attempted, such as allowing multiple values within the network or making them trainable. In this way neurons would specialize to their preferred time-scale but this would require further experiments.

The activity rate is properly regularized, preventing the neurons from spiking at high frequencies. In this way the sparsity and asynchronicity of the input is retained throughout the network. This is usually problematic in regular CNN, because, even if the input is sparse, as more and more convolutions are applied the sparsity is lost [144]. By keeping the network's activity as low as possible, DeCoLLe optimizes the amount of processing required resulting in reduced computation and power consumption.

Finally, DeCoLLe is directly portable to neuromorphic hardware, allowing for full exploitation of spikes and asynchronicity.

Chapter 6

Conclusions

This thesis work presented analysis and implementation of algorithms for object detection and recognition using event-based cameras. To address these tasks it was necessary to understand limits and advantages of the sensor, and the associated data representation, in the target robotic scenario. One of the big challenges posed by this specific scenario is the activity generated in cluttered scenes by the robot's motion.

This challenge was addressed with the development of an event-driven model of attention that specifically selects object-like stinuli as described in Chapter 3. To be relevant in a robotic application, the attention must select regions of the visual field which contain objects that iCub could interact with, therefore the system has been tailored to elicit a stronger response in presence of objects within a certain range of distances from the robot and sizes. The attention model was derived from biological evidence of similar attentive system in humans. In this sense, the use of event cameras represent an increase in the overall plausibility of the model, as demonstrated in Chapter 3, in which the sensor replaces some of the early computational stages of the model, while off-loading part of the processing from the software module.

The attention alone, however, does not provide a complete image processing system, as it represents one of the earliest stages of this pipeline. Once the object is detected, it needs to be recognized. To tackle the recognition task, a feasibility study was required to prove that the events carry sufficient information to distinguish different classes of objects, as further described in Chapter 4. The doubt was given by the significant difference between the appearance of frame and event-based cameras outputs. To dispel this doubt, an off-the-shelf state-of-the-art deep learning architecture was retrained with event data.

Unfortunately, as pointed out in Section 2.5, the small number of available datasets for event data represents a limitation when designing a spike-based algorithm. For this reason, Section 4.1.3 introduced an automatic annotation system to easily record labelled data, resulting in a dataset for object recognition with class and bounding box labels, where the head motion ensures the presence of background clutter.

Chapter 4 demonstrated that event-based object detection and recognition is at least possible for natural robotic scenes with background clutter using off-the-shelf deep learning. The conditions are somewhat more difficult than prior work in event-based CNNs [104, 129] and more similar to typical robot conditions.

To feed the CNN with an image, it was necessary to accumulate events over a certain temporal window. The problem with this kind of representation is that the rich temporal information contained in the events is lost and that a manual tuning of the temporal parameter is required. Thereby, an analysis on the tuning of that parameter has been carried out. Results indicate that performance is sensitive to the window duration which influences how long visual information is relevant, to within a *single order of magnitude*. The implication is that for very specific tasks the parameter may need to be tuned appropriately, but for a wide

range of tasks typical of a domestic robot, a single parameter should suffice. Nevertheless, results of experiments in very fast situations and low-light conditions demonstrated that the employment of event cameras has some advantages over the frame-based ones.

Results from Chapter 4 provided a proof-of-concept of usability of the events for the target application. However, the use of off-the-shelf architectures, which are not intended to process event data, prevents a full exploitation of the sensor capabilities. For this reason, Chapter 5 introduces the implementation of DeCoLLe, a SNN specifically designed to deal with spike-based data. DeCoLLe provides an architecture and learning rule which overcome some of the problems of BP which prevent the famous learning algorithm from being applied to SNN, neuromorphic hardware and being biologically plausible. In particular, BP works under the assumption of symmetric connectivity, alternation of forward and backward passes, and the use of precise communication channels and error signals. These have been relaxed using a three-factor learning rule which depends only on local variables and uses a random projection of the error signal, solving the weight transport, update-locking and non-locality problems introduced by BP. More details can be found in Sections 5.1 and 5.2 which provide a mathematical formulation of the neuron dynamics and learning rule respectively.
The implemented architecture is finally evaluated on an expanded version of the dataset recorded in Chapter 4 yielding promising results, even better than the ones obtained in Chapter 4, demonstrating that a properly trained spiking architecture is better than regular CNNs to deal with data coming from event cameras. Additionally, DeCoLLe is, by design, well suited for deployment on neuromorphic hardware, and therefore represents one further step towards the development of a fully asynchronous, low-latency and low power consumption image processing

pipeline.

6.1 Future works

An ongoing project is currently focusing on integrating the attention and recognition system in a fully spiking setting. To this aim a SpiNNaker implementation of the proto-object saliency model is being developed, which first requires the conversion of the model to a spiking setup. In parallel, the attention mechanism is being used as a preprocessing stage of the SNN recognition system, in order to deliver a full detection and recognition pipeline. The final goal of this project is the implementation of the whole pipeline in a neuromorphic, asynchronous fashion in a robot, which could naturally interact with the surroundings by intelligently exploring the visual field, selecting the most salient regions and subsequently recognize what the attended region contains. This thesis represents a small step towards the development of a robot which is more aware of its environment and that can explore and interact with it in an efficient manner. As such, there are many open questions and further developments that can be explored.

As an example, the kernels utilized in the attention module can be improved by applying learning technique. Some preliminary attempts in this direction have already been done, but further experiments are necessary. Additionally, in order to enable an active interaction, the robot could track previously selected regions and subsequently focus its attention towards new ones, i.e. by implementing an Inhibition Of Return (IOR) mechanism, which would result in an exploratory behaviour.
As far as the recognition is concerned, the DeCoLLe architecture could be exploited more, by harnessing the inherent temporal information induced by the recurrent structure of the network. In a previous work [20], DeCoLLe has been

utilized to recognize gestures which require to retain some temporal dependency in order to be properly classified. In future developments, the dataset utilized in this thesis could be expanded to contain i.e. actions, involving or not the presence of object, and use a similar architecture to distinguish them. Further experiments could also be done to test the network accuracy under different conditions of motion or illumination to fully capture the benefits of event sensors.

References

[1] Chiara Bartolozzi, Francesco Rea, Charles Clercq, Daniel B. Fasnacht, Giacomo Indiveri, Michael Hofstatter, and Giorgio Metta. Embedded neuromorphic vision for humanoid robots. In *CVPR 2011 WORKSHOPS*, number May 2014, pages 129–135. IEEE, jun 2011. 2, 41, 53

[2] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. The iCub humanoid robot: an open platform for research in embodied cognition. *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 50–56, 2008. 2, 41

[3] Alexander F. Russell, Stefan Mihalaş, Rudiger von der Heydt, Ernst Niebur, and Ralph Etienne-Cummings. A model of proto-object based saliency. *Vision Research*, 94:1–15, 2014. 3, 33, 41, 43, 44, 45, 46, 47, 48, 54, 59

[4] Massimiliano Iacono, Giulia D'Angelo, Arren Glover, Vadim Tikhanoff, Ernst Niebur, and Chiara Bartolozzi. Proto-object based saliency for event-driven cameras. *IROS 2019*, 2019. 4, 62

[5] Massimiliano Iacono, Stefan Weber, Arren Glover, and Chiara Bartolozzi. Towards Event-Driven Object Detection with Off-The-Shelf Deep Learning. In *IEEE International Conference on Intelligent Robots and Systems*. IROS 2018, 2018. 5, 41, 53, 56

[6] Y Lecun, Y Bengio, and G Hinton. Deep learning. *Nature*, 2015. 7, 34, 63

[7] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 × 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. 7, 11, 32

[8] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240 × 180 130 dB 3 μs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, oct 2014. 7, 12

[9] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS. *IEEE Journal of Solid-State Circuits*, 46(1):259–275, jan 2011. 7, 12, 41, 53

[10] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, dec 1997. 8, 16, 34

[11] John Backus. Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs. *Communications of the ACM*, 21(8):613–641, aug 1978. 9

[12] Misha A Mahowald and Carver Mead. The silicon retina, 1991. 10, 11

[13] Alex Lubben. Self-driving Uber killed a pedestrian as human safety driver watched, 2018. 11

[14] Marc Osswald, Sio-Hoi Hoi Ieng, Ryad Benosman, and Giacomo Indiveri. A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems. *Scientific Reports*, 7:40703, jan 2017. 11

[15] Zhen Xie, Shengyong Chen, and Garrick Orchard. Event-based stereo depth estimation using belief propagation. *Frontiers in Neuroscience*, 11(OCT):535, oct 2017. 11, 27

[16] Germain Haessig, Xavier Berthelon, Sio Hoi Ieng, and Ryad Benosman. A Spiking Neural Network Model of Depth from Defocus for Event-based Neuromorphic Vision. *Scientific Reports*, 9(1):3744, dec 2019. 11

[17] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018. 11, 28

[18] Enrico Calabrese, Gemma Taverni, Christopher Awai Easthope, Sophie Skriabine, Federico Corradi, Luca Longinotti, Kynan Eng, and Tobi Delbruck. DHP19: Dynamic Vision Sensor 3D Human Pose Dataset. In *CVPR*, 2019. 11, 14, 28

[19] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, Dharmendra Modha, U C San Diego, and Uzheth Zurich. A Low Power, Fully Event-Based Gesture Recognition System. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017-Janua:7388–7397, jul 2017. 11, 27, 29

[20] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic Plasticity Dynamics for Deep Continuous Local Learning. 2018. 11, 20, 83, 86, 89, 99

[21] Jean-Matthieu Maro and Ryad Benosman. Event-based Gesture Recognition with Dynamic Background Suppression using Smartphone Computational Capabilities. nov 2018. 11, 15

[22] Bibrat Ranjan Pradhan, Yeshwanth Bethi, Sathyaprakash Narayanan, Anirban Chakraborty, and Chetan Singh Thakur. N-HAR: A Neuromorphic Event-Based Human Activity Recognition System using Memory Surfaces. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, may 2019. 11

[23] Henri Rebecq, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza. EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017. 11

[24] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9910 LNCS, pages 349–364. 2016. 11

[25] P.-F. Ruedi, Pascal Heim, François Kaess, Eric Grenet, Friedrich Heitger, P.-Y. Burgi, Steve Gyger, and Pascal Nussbaum. A 128 × 128 pixel 120 dB dynamic range vision sensor chip for image contrast and orientation extraction. In *2003 IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC.*, volume 1, pages 226–490. IEEE, 2003. 11

[26] Kareem A. Zaghloul and Kwabena Boahen. Optic Nerve Signals in a Neuromorphic Chip I: Outer and Inner Retina Models. *IEEE Transactions on Biomedical Engineering*, 51(4):657–666, apr 2004. 11

[27] P. Lichtsteiner and T. Delbruck. A 64×64 AER logarithmic temporal derivative silicon retina. In *2005 PhD Research in Microelectronics and Electronics - Proceedingsof the Conference*, volume II, pages 406–409. IEEE, 2005. 11

[28] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128 × 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. In *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, pages 2060–2069. IEEE, 2006. 11

[29] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. A 128,× 128 1.5% contrast sensitivity 0.9% FPN 3 μs latency 4 mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits*, 48(3):827–838, 2013. 11

[30] Raphael Berner, Christian Peter Brändli, Minhao Yang, Shih-Chii Liu, and Tobi Delbrück. A 240×180 120dB 10mW 12us - latency sparse output vision sensor for mobile applications. *International image sensor workshop (IISW)*, pages 186–187, 2013. 12

[31] iniVation — Neuromorphic vision systems. 12

[32] Yoel Yaffe, Nathan Levy, Evgeny Soloveichik, Sebastien Derhy, Ayal Keisar, Elad Rozin, Liron Artsi Jun-Seok Kim, Keunju Park, Bongki Son, Yunjae Suh, Heejae Jung, Changwoo Shin, Jooyeon Woo, Yohan Roh, Hyunku Lee, and Hyunsurk Ryu. ICRA'17 Workshop on Event-based Vision; Dynamic Vision Sensor. Technical report, 2017. 12

[33] Insightness AG. Insightness – Sight for your device, 2018. 12

[34] SmartThings Vision — GP-U999GTEEAAC — Samsung AU. 12

[35] Prophesee - Metavision for Machines. page 14, 2006. 13

[36] Alessandro Mortara. A Pulsed Communication/Computation Framework for Analog VLSI Perceptive Systems. In T S Lande, editor, *Neuromorphic Systems Engineering*, pages 201–215. Springer US, Boston, MA, 1998. 13

[37] José Antonio Pérez-Carrasco, Bo Zhao, Carmen Serrano, Begoña Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabé Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing - Application to feedforward convnets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2706–2719, nov 2013. 14, 18, 35, 40, 90

[38] Rohan Ghosh, Tang Siyi, Mahdi Rasouli, Nitish V. Thakor, and Sunil L. Kukreja. Pose-Invariant Object Recognition for Event-Based Vision with Slow-ELM. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9887 LNCS, pages 455–462. Springer International Publishing, 2016. 14, 27, 36, 40

[39] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. Events-to-Video: Bringing Modern Computer Vision to Event Cameras. 2019. 14

[40] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E. Shi, and Ryad B. Benosman. HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1346–1359, jul 2017. 14, 17, 18

[41] Saeed Afshar, Tara Julia Hamilton, Jonathan Tapson, André Van Schaik, and Gregory Cohen. Investigation of event-based surfaces for high-speed detection, unsupervised feature extraction, and object recognition. *Frontiers in Neuroscience*, 13(JAN):1047, jan 2019. 15, 18, 27

[42] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Asynchronous Convolutional Networks for Object Detection in Neuromorphic Cameras. 2018. 15, 27, 36, 40

[43] Anton Mitrokhin, Cornelia Fermuller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based Moving Object Detection and Tracking. In *IROS 2018*, pages 6895–6902, 2018. 16, 18

[44] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised Event-based Learning of Optical Flow, Depth, and Egomotion. 2018. 16, 18

[45] Bodo Rückauer, Nicolas Känzig, Shih-Chii Liu, Tobi Delbruck, and Yulia Sandamirskaya. Closing the Accuracy Gap in an Event-Based Visual Recognition Task. 2019. 16, 18, 35

[46] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Attention mechanisms for object recognition with event-based cameras. In *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, pages 1127–1136, 2019. 18

[47] W. Maass and H. Markram. On the computational power of circiuts of spiking neurons. *Journal of Computer and System Sciences*, 69(4):593–616, 2004. 17

[48] B. Segee. Methods in Neuronal Modeling: from Ions to Networks, 2nd Edition. *Computing in Science & Engineering*, 1(1):81–81, jan 1999. 18

[49] Evangelos Stromatias, Miguel Soto, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. An Event-Driven Classifier for Spiking Neural Networks Fed with Synthetic or Dynamic Vision Sensor Data. *Frontiers in Neuroscience*, 11(JUN):1–17, jun 2017. 18, 35

[50] Diederik Paul Moeys, Federico Corradi, Emmett Kerr, Philip Vance, Gautham Das, Daniel Neil, Dermot Kerr, and Tobi Delbruck. Steering a predator robot using a mixed frame/event-driven convolutional neural network. In *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–8. IEEE, jun 2016. 18

[51] Emre O. Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in Neuroscience*, 11(JUN):1–18, 2017. 19, 79, 82

[52] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random feedback weights support learning in deep neural networks. *Nature Publishing Group*, 7:1–10, 2016. 19, 79, 81, 82

[53] Y. Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 19, 27

[54] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. EMNIST: Extending MNIST to handwritten letters. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2017-May, pages 2921–2926, 2017. 19

[55] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate Gradient Learning in Spiking Neural Networks. jan 2019. 20, 21, 79, 83, 88

[56] Hesham Mostafa, Vishwajith Ramesh, and Gert Cauwenberghs. Deep supervised learning using local errors. *Frontiers in Neuroscience*, 12(AUG):1–16, 2018. 20, 81, 83, 87

[57] Yang Dan and Mu Ming Poo. Spike timing-dependent plasticity of neural circuits, 2004. 20

[58] Johannes C Thiele, Olivier Bichler, and Antoine Dupret. Event-Based, Timescale Invariant Unsupervised Online Deep Learning With STDP. *Frontiers in Computational Neuroscience*, 12:46, 2018. 20, 35, 40

[59] Maurizio Valle, DanieleD. Caviglia, and GiacomoM. Bisio. An experimental analog VLSI neural network with on-chip back-propagation learning. *Analog Integrated Circuits and Signal Processing*, 9(3):203–206, apr 1996. 22

[60] J.G. Eldredge and B.L. Hutchings. Density enhancement of a neural network using FPGAs and run-time reconfiguration. In *Proceedings of IEEE Workshop on FPGA's for Custom Computing Machines*, number 2, pages 180–188. IEEE Comput. Soc. Press, 1994. 22

[61] J Ouali and G Saucier. Fast Generation of Neuro-ASICs. In *International Neural Network Conference*, pages 563–567. Springer Netherlands, Dordrecht, 1990. 22

[62] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R Chandrasekaran, Jean Marie Bussat, Rodrigo Alvarez-Icaza, John V Arthur, Paul A Merolla, and Kwabena Boahen. Neurogrid: A

mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014. 22

[63] Stefan Scholze, Stefan Schiefer, Johannes Partzsch, Stephan Hartmann, Christian Georg Mayr, Sebastian Höppner, Holger Eisenreich, Stephan Henker, Bernhard Vogginger, and Rene Schüffny. VLSI implementation of a 2.8 Gevent/s packet-based AER interface with routing and event sorting functionality. *Frontiers in Neuroscience*, 5(OCT):1–13, 2011. 22

[64] Stefan Scholze, Holger Eisenreich, Sebastian Höppner, Georg Ellguth, Stephan Henker, Mario Ander, Stefan Hänzsche, Johannes Partzsch, Christian Mayr, and René Schüffny. A 32 GBit/s communication SoC for a waferscale neuromorphic system. *Integration, the VLSI Journal*, 45(1):61–75, 2012. 22

[65] Sebastian Millner, Andreas Grübl, Karlheinz Meier, Johannes Schemmel, and Marc Olivier Schwartz. A VLSI implementation of the adaptive exponential integrate-and-fire neuron model. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, NIPS 2010*, 2010. 22

[66] Johannes Schemmel, Daniel Briiderle, Andreas Griibl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1947–1950. IEEE, may 2010. 22

[67] Steve B. Furber, Francesco Galluppi, Steve Temple, and Luis A. Plana. The SpiNNaker Project. *Proceedings of the IEEE*, 102(5):652–665, may 2014. 23

[68] Evangelos Stromatias, Cameron Patterson, and Steve Furber. Optimising the overall power usage on the SpiNNaker neuromimetic platform. In *Proceedings of the International Joint Conference on Neural Networks*, pages 4280–4287. Institute of Electrical and Electronics Engineers Inc., sep 2014. 23

[69] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, aug 2014. 23

[70] Jeremy Hsu. IBM's new brain [News]. *IEEE Spectrum*, 51(10):17–19, sep 2014. 23

[71] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Frontiers in Neuroscience*, 9(APR):1–17, apr 2015. 24

[72] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro*, 38(1):82–99, jan 2018. 25

[73] Tensor processing unit, 2019. 25

[74] Google. Dev Board — Coral, 2019. 25

[75] Asus. Asus Tinker Board S, 2019. 25

[76] Esin Yavuz, James Turner, and Thomas Nowotny. GeNN: A code generation framework for accelerated brain simulations. *Scientific Reports*, 6, 2016. 25

[77] James C. Knight and Thomas Nowotny. GPUs Outperform Current HPC and Neuromorphic Solutions in Terms of Speed and Energy When Simulating a Highly-Connected Cortical Model. *Frontiers in Neuroscience*, 12(December):1–19, dec 2018. 26

[78] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 26

[79] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):740–755, 2014. 26, 71

[80] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 26

[81] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience*, 9(NOV):1–11, nov 2015. 27

[82] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, apr 2006. 27

[83] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. Poker-DVS and MNIST-DVS. Their history, how they were made, and other details. *Frontiers in Neuroscience*, 9(DEC):1–10, dec 2015. 27, 29, 35

[84] Guang Chen, Jieneng Chen, Marten Lienen, Jörg Conradt, Florian Röhrbein, and Alois C. Knoll. FLGR: Fixed length GISTS representation learning for RNN-HMM hybrid-based neuromorphic continuous gesture recognition. *Frontiers in Neuroscience*, 13(FEB):73, feb 2019. 27

[85] Anton Mitrokhin, Chengxi Ye, Cornelia Fermuller, Yiannis Aloimonos, and Tobi Delbruck. EV-IMO: Motion Segmentation Dataset and Learning Pipeline for Event Cameras. (August), mar 2019. 28

[86] S. P. Strong, Roland Koberle, Rob R. de Ruyter van Steveninck, and William Bialek. Entropy and Information in Neural Spike Trains. *Physical Review Letters*, 80(1):197–200, jan 1998. 29

[87] Kristin Koch, Judith McLean, Michael Berry, Peter Sterling, Vijay Balasubramanian, and Michael A. Freed. Efficiency of Information Transmission by Retinal Ganglion Cells. *Current Biology*, 14(17):1523–1530, sep 2004. 29

[88] Christof Koch and Shimon Ullman. Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry. In *Matters of Intelligence*, pages 115–141. Springer Netherlands, Dordrecht, 1987. 30

[89] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998. 30, 43, 52

[90] Laurent Itti and Christof Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10-12):1489–1506, 2000. 30

[91] Laurent Itti. Feature combination strategies for saliency-based visual attention systems. *Journal of Electronic Imaging*, 10(1):161, jan 2001. 30

[92] Laurent Itti. Quantifying the contribution of low-level saliency to human eye movements in dynamic scenes. *Visual Cognition*, 12(6):1093–1123, 2005. 30

[93] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, mar 2001. 30, 31, 32

[94] Raymond M. Klein. Inhibition of return. *Trends in Cognitive Sciences*, 4(4):138–147, 2000. 31

[95] Ruth Kimchi, Yaffa Yeshurun, and Aliza Cohen-Savransky. Automatic, stimulus-driven attentional capture by objecthood. *Psychonomic Bulletin & Review*, 14(1):166–172, feb 2007. 31, 32, 33, 54

[96] Ronald A. Rensink. The Dynamic Representation of Scenes. *Visual Cognition*, 7(1-3):17–42, jan 2000. 31, 43

[97] Wolfgang Köhler. Gestalt psychology. *Psychological research*, 31(1):XVIII—-XXX, 1967. 32, 44, 45

[98] Francesco Rea, Giorgio Metta, and Chiara Bartolozzi. Event-driven visual attention for the humanoid robot iCub. *Frontiers in Neuroscience*, 7(7 DEC), 2013. 32

[99] Xin Yang, Yanpei Zhou, Dake Zhou, and Yinji Hu. Image segmentation and proto-objects detection based visual tracking. *Optik - International Journal for Light and Electron Optics*, 131:1085–1094, 2017. 33

[100] Dirk Walther and Christof Koch. Modeling attention to salient proto-objects. *Neural Networks*, 19(9):1395–1407, nov 2006. 33

[101] Hong Zhou, Howard S. Friedman, and Rüdiger von der Heydt. Coding of border ownership in monkey visual cortex. *The Journal of Neuroscience*, 20(17):6594–6611, sep 2000. 33, 50

[102] Jonathan Williford and Rudiger Heydt. Border-ownership coding. *Scholarpedia*, 8(10):30040, 2013. 33

[103] Himanshu Akolkar, Cedric Meyer, Xavier Clady, Olivier Marre, Chiara Bartolozzi, Stefano Panzeri, and Ryad Benosman. What Can Neuromorphic Event-Driven Precise Timing Add to Spike-Based Pattern Recognition? *Neural Computation*, 27(3):561–593, mar 2015. 34

[104] Peter U. Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. *Proceedings of the International Joint Conference on Neural Networks*, 2015-Septe:1–8, jul 2015. 35, 97

[105] Mazdak Fatahi, Mahmood Ahmadi, Mahyar Shahsavari, Arash Ahmadi, and Philippe Devienne. evt_MNIST: A spike based version of traditional MNIST. apr 2016. 35

[106] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. HFirst: A Temporal Approach to Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2028–2040, 2015. 35, 40

[107] D H Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154, jan 1962. 35, 42

[108] Bharath Ramesh, Andrés Ussa, Luca Della Vedova, Hong Yang, and Garrick Orchard. PCA-RECT: An Energy-Efficient Object Detection Approach for Event Cameras. *LNCS*, 11367:434–449, 2018. 36, 37, 40

[109] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002. 36

[110] Guang Bin Huang, Qin Yu Zhu, and Chee Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006. 36

[111] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 779–788. IEEE, jun 2016. 36

[112] Freiburg Im Breisgau, June Ee, Alessandro Zanardi, Andreas Aumiller, Julian Zilly, Andrea Censi, and Emilio Frazzoli. Cross-Modal Learning

Filters for RGB-Neuromorphic Wormhole Learning. In *Robotics: Science and Systems XV*, 2019. 37, 40

[113] Freiburg Im Breisgau, June Ee, Alessandro Zanardi, Andreas Aumiller, Julian Zilly, Andrea Censi, and Emilio Frazzoli. Cross-Modal Learning Filters for RGB-Neuromorphic Wormhole Learning. In *Robotics: Science and Systems XV*, 2019. 38

[114] Giulia Pasquale, Carlo Ciliberto, Lorenzo Rosasco, and Lorenzo Natale. Object identification from few examples by improving the invariance of a Deep Convolutional Neural Network. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2016-Novem, pages 4904–4911. IEEE, oct 2016. 38, 70

[115] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4144–4149. IEEE, oct 2016. 38

[116] Tobi Delbruck and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience*, 7:223, nov 2013. 38

[117] Arren Glover and Chiara Bartolozzi. Robust visual tracking with a freely-moving event camera. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776. IEEE, sep 2017. 38

[118] Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. EMVS : Event-based Multi-View Stereo. *Bmvc*, pages 1–11, 2016. 38

[119] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime estimation of events from Dynamic Vision Sensors. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2015-June, pages 4874–4881, 2015. 40

[120] Stephen W. Kuffler. Discharge Patterns and Functional Organization of Mammalian Retina. *Journal of Neurophysiology*, 16(1):37–68, jan 1953. 43

[121] Edgar Rubin. Visuell Wahrgenommene Figuren [Visually perceived patterns], 1921. 45

[122] Nikos K. Logothetis, David A. Leopold, and David L. Sheinberg. What is rivalling during binocular rivalry? *Nature*, 380(6575):621–624, apr 1996. 46

[123] E. Craft, H. Schutze, E. Niebur, and R. von der Heydt. A Neural Model of Figure-Ground Organization. *Journal of Neurophysiology*, 97(6):4310–4326, 2007. 51

[124] Giulia Vezzani, Ugo Pattacini, and Lorenzo Natale. A grasping approach based on superquadric models. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1579–1586. IEEE, may 2017. 53

[125] Lorenzo Jamone, Alexandre Bernardino, and Jose Santos-Victor. Benchmarking the Grasping Capabilities of the iCub Hand With the YCB Object and Model Set. *IEEE Robotics and Automation Letters*, 1(1):288–294, jan 2016. 53

[126] Ali Borji and Laurent Itti. CAT2000: A Large Scale Fixation Dataset for Boosting Saliency Research. (November), may 2015. 55, 56

[127] Stefan Mihalas, Yi Dong, R. von der Heydt, and Ernst Niebur. Mechanisms of perceptual organization provide auto-zoom and auto-localization

for attention to objects. *Proceedings of the National Academy of Sciences*, 108(18):7583–7588, may 2011. 54

[128] Y Kim, W Jung, and H Bang. Visual Target Tracking and Relative Navigation for Unmanned Aerial Vehicles in a GPS-Denied Environment. *International Journal of Aeronautical and Space Sciences*, 15(3):258–266, 2014. 63

[129] Rohan Ghosh, Abhishek Mishra, Garrick Orchard, and Nitish V. Thakor. Real-time object recognition and orientation estimation using an event-based camera and CNN. In *IEEE 2014 Biomedical Circuits and Systems Conference, BioCAS 2014 - Proceedings*, pages 544–547, 2014. 63, 77, 97

[130] Himanshu Akolkar, David Reverter Valeiras, Ryad Benosman, and Chiara Bartolozzi. Visual-auditory saliency detection using event-driven visual sensors. In *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–6. IEEE, jun 2015. 63

[131] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS:21–37, 2016. 65, 66, 70

[132] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. pages 7310–7319, nov 2016. 66, 68

[133] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE, jun 2015. 70

[134] Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541, jun 2018. 79

[135] Pierre Baldi, Peter Sadowski, and Zhiqin Lu. Learning in the Machine: Random Backpropagation and the Learning Channel. pages 1–57, 2016. 79

[136] Pierre Baldi and Peter Sadowski. A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Networks*, 83:51–74, nov 2016. 79

[137] Robert Urbanczik and Walter Senn. Learning by the Dendritic Prediction of Somatic Spiking. *Neuron*, 81(3):521–528, 2014. 79, 82

[138] Qianli Liao, Joel Z. Leibo, and Tomaso Poggio. How important is weight symmetry in backpropagation? *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 1837–1844, 2016. 79, 80

[139] Emre O. Neftci. Data and Power Efficient Intelligence with Neuromorphic Learning Machines. *iScience*, 5:52–68, jul 2018. 79

[140] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 80

[141] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics*. Cambridge University Press, Cambridge, 2014. 85

[142] Ronald J Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 1989. 86

[143] Melika Payvand, Mohammed Fouda, Fadi Kurdahi, Ahmed Eltawil, and Emre O Neftci. Error-triggered Three-Factor Learning Dynamics for Crossbar Arrays. 2019. 90

[144] Benjamin Graham and Laurens van der Maaten. Submanifold Sparse Convolutional Networks. pages 1–10, 2017. 95

Acronyms

AER Address Event Representation. 13, 35

ANN Artificial Neural Network. 8, 9, 16, 18, 19, 35, 36, 90

API Application Programming Interface. 67, 69

ASIC Application Specific Integrated Circuits. 22

ATIS Asynchronous Time-based Image Sensor. 7, 12, 17, 65, 67

BP Back Propagation. 5, 9, 79, 80, 82, 94, 98

CISC Complex Instruction Set Computer. 25

CMOS Complementary metal oxide semiconductor. 65

CNN Convolutional Neural Network. 8, 16, 19, 35, 39, 64–67, 73, 78, 95, 97, 98

CPU Central Processing Unit. 22

CS Center Surround. ix, x, 3, 42–45, 47, 48, 61

DeCoLLe Deep Continuous Local Learning. xi, 5, 83, 86, 87, 89, 90, 93–95, 98, 99

DVS Dynamic Vision Sensor. ix, 7, 11, 12, 19, 27, 28, 36

DYNAP Dynamic Neuromorphic Asynchronous Processor. 25, 26

ELM Extreme Learning Machine. 40

eRBP Event Driven Random Backpropagation. 19, 20, 82, 83

FA Feedback Alignment. xi, 81, 82

FIFO First In First Out. 40

GPU Graphics Processing Unit. 9, 22, 26, 39, 72, 89, 93

HICANN High Input Count Analog Neural Network. 23

IMU Inertial Measurement Unit. 28

IOR Inhibition Of Return. 31, 32, 99

IOU Intersection Over Union. 74, 75

IROS International Conference on Intelligent Robots and Systems. 4, 5

LIF Leaky Integrate and Fire. 19, 25, 40, 83

LTD Long Term Depression. 21

LTP Long Term Potentiation. 21, 24

mAP Mean Average Precision. 40

MSE Mean Squared Error. 91

MVSEC Multi Vehicle Stereo Event Camera. 28

NMI Neuromorphic Machine Intelligence. 5

NN Neural Network. 40, 80

PCA Principal Component Analysis. 36–38, 40

PSP Post Synaptic Potential. 84

RGB Red Green Blue. x, 1, 2, 4, 5, 8, 13, 27, 34, 37, 38, 40, 43, 45, 57, 68, 97

RISC Reduced Instruction Set Computer. 25

RNN Recurrent Neural Network. 85, 86, 93

ROLLS Reconfigurable On-Line Learning Spiking. 24–26

RTRL Real Time Recurrent Learning. 86

SFA Slow Feature Analysis. 40

SLAM Simultaneous Localization and Mapping. 39

SNN Spiking Neural Network. 5, 8, 9, 16, 18–21, 25, 26, 35, 36, 40, 66, 79, 85, 90, 91, 93, 98, 99

SRM Spike Response Model. 85

SSD Single Shot Detector. 66, 71, 72

STDP Spike Timing Dependent Plasticity. 20, 21, 35, 36

STP Short Term Potentiation. 24

TPU Tensor Processing Unit. 25, 26

VLSI Very Large Scale Integration. 24

VM Von Mises. 44, 49, 50, 54, 58, 61

WTA Winner Take All. 40

YOLE You Only Look Events. 37

YOLO You Only Look Once. 37, 40